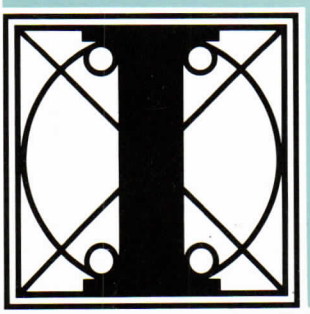
Operating System

SHELL Commands

Reference Manual

MOS



olivetti

PUBLICATION ISSUED BY:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 IVREA (Italy)

Information from Olivetti Documentation

Operating System

SHELL Commands

Reference Manual

olivetti

PREFACE

This manual describes the control of the operating system MOS, in the SHELL environment. It provides information on keyboard handling, input entry and output control, and details on the procedures and commands essential to process control. It also describes how a user may configure the system to cater especially for his own requirements.

Required reading for the best use of the information in this manual is Introduction to MOS (for a description of files, directories, volumes, and devices), and MCL MOS Command Language User Guide, (for background to the language).

SUMMARY

This manual is organised thus:

- Part 1 describes the Shell environment, how to input instructions to it, its operation, and the main characteristics of MCL.
- Part 2 details the Shell commands available to all users, in alphabetical order.
- Part 3 details the Shell commands reserved for the system administrator, also in alphabetical order.
- Part 4 contains all the appendices:
  - Appendix A is a list of all system-produced error messages, their meaning, and action required.
  - Appendix B details the limits of hardware and software objects significant in the Shell environment.
  - Appendix C provides detailed and background information about magnetic tapes and their use.
  - Appendix D provides detailed and background information about floppy discs and their use.

REFERENCES

Read first ...

Introduction to MOS (in MOS Features and Functions - Code 4041600 F)

Further information is provided in ...

DOCUMENTATION Guide - Code 4041610 U

ENCRYPTION Driver Interface Programmer Guide - Code 4004240 U

Glossary/Glossario - Code 4002140 H

MCL MOS Command Language Pocket Reference - Code 4002670 N

MCL MOS Command Language User Guide
(in SHELL Environment User Manual - Code 4041630 W)

MOS Distributed System in Local Area Network (LAN) User Guide -
Code 4042160 T

MOS Operating Guide - Code 4036160 T

MOS Programmer Guide - Code 4041670 S

MOS System Software Generation and Installation Manual -
Code 4041710 W

MOS System Software Maintenance Tools User Guide - Code 4041730 Y

FIRST EDITION: January 1988 - Release 6.0

CONTENTS

PAGE

0-1 SHELL COMMANDS INDICES

0-3 SHELL FUNCTIONS COMMAND PERMUTED INDEX

0-9 SHELL COMMANDS FUNCTION PERMUTED INDEX

PART 1 - THE SHELL ENVIRONMENT

INTRODUCTION TO PART 1

1-1 1. GENERAL CHARACTERISTICS

1-1 THE VIDEO

1-1 USER VIDEO

1-2 SYSTEM LINE

1-2 FULL SCREEN VIDEO

1-2 THE KEYBOARD

1-4 MASTER WORKSTATION

1-4 WARNING TERMINAL

1-5 MASTER TERMINAL

1-7 LOCAL AND GLOBAL MWS

1-7 LOADING AND EXECUTING COMMANDS

1-7 VIDEO INPUT LINE

1-7 SYSTEM LINE INPUT

1-8 COMMAND PARAMETERS

1-8 OPTIONS

1-9 INTERACTIVE COMMANDS

1-9 ASYNCHRONOUS CONTROL COMMANDS

1-9 MCL COMMAND EXECUTION IN ASYNCHRONOUS MODE

PAGE

1-9 BUILT-IN COMMANDS

1-10 REMOTE COMMANDS

1-10 COMPLETION CODE

1-10 REDIRECTION OF INPUT AND OUTPUT

1-11 ERROR HANDLING

1-11 COMMAND REPETITION

1-12 COMMAND EDITING

1-13 .MESG FILE

1-14 .INIT FILE

1-15 OPERATING PROCEDURES

1-15 LOGIN AND LOGOUT

1-15 SYSTEM ADMINISTRATOR FUNCTIONS

1-15 PASSWORD

2-1 2. SPOOL SYSTEM AND BATCH MANAGEMENT

2-1 THE SPOOL SYSTEM

2-1 SPOOL OBJECTS

2-2 OTHER UNSPOOLER INFORMATION

2-2 LINKS BETWEEN SPOOL OBJECTS

2-3 CONTROL

2-3 PRINTING PROGRAM FILES THROUGH SPOOL

2-4 DISTRIBUTED CONFIGURATION

2-4 BATCH MANAGEMENT

2-4 SUBMITTING JOBS

2-4 USER ACCESS

2-5 SHELL VARIABLES

2-5 DISTRIBUTED CONFIGURATION

PART 2 - COMMANDS

PAGE

INTRODUCTION TO PART 2

3-1 3. DESCRIPTION OF THE SHELL COMMANDS

3-1 INTRODUCTION

3-1 COMMAND SYNTAX

3-1 SYMBOLS USED IN THE EXAMPLES

PART 3 - RESERVED COMMANDS

INTRODUCTION TO PART 3

4-1 4. RESERVED COMMANDS DESCRIPTION

4-1 INTRODUCTION

PART 4 - APPENDICES

INTRODUCTION TO PART 4

A-1 A. ERROR MESSAGES

A-2 SYSTEM ERRORS

A-13 BATCH ERROR MESSAGES

A-14 SPOOL ERROR MESSAGES

A-17 MOS COMMANDS ERROR MESSAGES

B-1 B. GENERAL LIMITATIONS AND CHARACTERISTICS

C-1 C. MAGNETIC TAPE CHARACTERISTICS

C-1 INTRODUCTION

C-1 TAPE TYPES

C-1 CONTENTS COMMON TO BOTH TYPES

C-1 TAPE MARKS AND LABELS

C-3 TAPE HEADER LABEL

C-3 VOLUME HEADER LABEL

C-4 TAPE DESCRIPTIONS

C-4 SCT (STREAMING CARTRIDGE TAPE)

C-6 MTU (MAGNETIC TAPE UNIT)

PAGE

C-6 TAPE OPERATING SPEEDS

D-1 D. FLOPPY DISC CHARACTERISTICS

D-1 INTRODUCTION

D-1 FLOPPY DISC DESCRIPTIONS

D-2 8 INCHES DISCS

D-2 5 INCHES DISCS

D-2 DISC ORGANISATION AND LABELS

D-2 INITIALISATION

D-3 CYLINDER 0 (INDEX CYLINDER)

D-4 TAPE MARK

D-4 OTHER CYLINDERS

D-4 RECURSIVE ORGANISATION

D-5 MULTI-FLOPPY ORGANISATION

D-6 DISPLAY OF CONTENTS

D-7 DISC STANDARDS AND COMPATIBILITY

SHELL COMMANDS INDICES

The following pages contain two indices, providing easy access to the SHELL command descriptions.

The first one, "Functions Command Index", lists the commands and their functions in order of command name.

The second one, "Commands Function Index", allows access to the commands by way of their associated function. The functions are listed by means of all the significant words appearing in the descriptions of the first index, ordered alphabetically.

The location of each command in the manual is given by its part number, followed by its chapter number.

SHELL FUNCTIONS COMMAND PERMUTED INDEX

"*": command used only by the system administrator.
"**": command found in the manual MCL MOS Command Language User Guide.

COMMAND	DESCRIPTION	Part/Chapter
!	repeat commmand	1/1
!!	edit command	1/1
BATCH	allows normal user some control of own jobs	2/3
BATCH	allows sys. admin. control of batch environment *	3/4
BM	executes a command in batch	**
BMQ	displays the batch classes	2/3
CALLMWS	sends a request to the Master WorkStation	2/3
CHTYPE	performs directory-program directory conversion	2/3
CLEAR	clears the whole screen	2/3
CLEARDIR	deletes the contents of a directory	2/3
CMPK	compacts a keyed file	2/3
COMPACT	compacts a disc file	2/3
CONN	connects a local name to a global name	**
CONSP	connects a local name to a SPOOL class	**
CONVERT	converts the type of a file	2/3
CONVPACK	converts a packed positional file for release 6.0	2/3
COPY	copies a file, directory, or volume	2/3
CPTREE	copies a directory or volume in recursive mode	2/3
DELINDEX	deletes an index from a keyed file	2/3
DEVSHARE	shares printer or cash adapter between 2 workstations *	3/4
DIFF	compares files or devices	2/3
DISCON	disconnects a local name	**
ECHO	displays the result of an expression	**

(cont.)

COMMAND	DESCRIPTION	Part/Chapter
EDPARSE	checks the syntax of an MCL procedure	**
EXIST	checks if a file exists	2/3
FCU	performs standard data conversion on floppy disc	2/3
FCUTAB	creates a user conversion table for use by FCU	2/3
FFT	allows file transfer from S6000 to L1 MOS system	2/3
FILETAR	dumps a file, directory, or volume onto MTU	2/3
FINGER	displays user information	2/3
FLD	dumps a file, directory, or volume onto floppy disc	2/3
FORMAT	converts a file from a 5.2 to a 6.0 file format	2/3
HELP	displays the Shell commands help file	2/3
INSTALPKG	installs or uninstalls a MOS package *	3/4
KILL	aborts a running user job	2/3
KILLFAM	aborts a family *	3/4
LASTCOM	displays the History List or modifies its size	**
LENGTH	calculates the length of a given string	**
LIST	displays the contents of a given file	2/3
LOGOUT	exits from the Shell environment	**
LPQ	displays printer and spooling class (job queue) status	2/3
LPR	prints files	2/3
LS	displays information on a file, directory, or volume	2/3
MHNAME	displays the machine name	2/3
MKALIAS	creates an alias file	2/3
MKBST	creates a byte-stream file	2/3
MKCEM	manages the centralised system messages database *	3/4
MKDIR	creates a directory	2/3

(cont.)

COMMAND	DESCRIPTION	Part/Chapter
MKENV	initialises track 0 of a floppy disc *	3/4
MKINDEX	creates a secondary index in a keyed file	2/3
MKKEYED	creates a keyed file	2/3
MKPOS	creates a positional file	2/3
MKVOL	creates a volume or allocates space for a memory dump	2/3
MNT	connects a physical volume on a drive to the system	2/3
MORE	displays a byte-stream file page by page	2/3
MTA	reads the messages sent to the Master WorkStation *	3/4
PACK	packs a byte-stream file	2/3
PRIOR	decreases the priority of all subsequent programs	**
PRY	displays information on a file, directory, or volume	2/3
RCODE	displays software release information	2/3
REMOVE	deletes files, directories, or volumes	2/3
RENAME	changes the name of a file, directory, or volume	2/3
RESUME	resumes a suspended user program	2/3
RIGHTS	reads or modifies the default access rights of new files	**
RP	displays the PDD physical description of a file	2/3
SETBUF	assigns private buffers to a file	2/3
SETDATE	sets or modifies the system date and time *	3/4
SETIDEN	changes the owner of a file, directory, or volume *	3/4
SETINFO	modifies the attributes of a given file	2/3
SETMODE	changes the access rights of a file system object	2/3
SETPASSW	creates or modifies a password	2/3
SETTERM	sets the screen size	2/3
SETWDIR	sets the working directory	**

(cont.)

COMMAND	DESCRIPTION	Part/Chapter
SHALIAS	displays the contents of an alias file	2/3
SHCON	displays the local and global names connected	**
SHDATE	displays the system date and time	2/3
SHDIR	list the contents of a directory	2/3
SHFAM	displays family status information	2/3
SHLABEL	displays the descriptor of a main volume	2/3
SHNAME	displays the user's login name	**
SHOW	displays the result of an expression	2/3
SHTERM	displays the current terminal characteristics	2/3
SHVAR	displays the value of an MCL variable	**
SHWDIR	displays the path name of the working directory	**
SPOOL	allows normal user control of own spooled jobs	2/3
SPOOL	allows sys. admin. control of spooling environment *	3/4
SUSPEND	suspends the job in execution	2/3
TAMROF	converts a file from a 6.0 to a 5.2 file format	2/3
TCU	performs standard data conversion on MTU	2/3
TRACE	creates a trace file for an MCL procedure	**
UNMNT	releases a connection between system and volume	2/3
UNPACK	unpacks a byte-stream file	2/3
USERMAN	allows normal user to list registered users and groups	2/3
USERMAN	allows sys. admin. to manage system users and groups *	3/4
VCOMPACT	compacts a volume *	3/4
VOLSR	dumps a volume onto SCT	2/3
WHEREIS	searches for a given file, directory, or volume	2/3
WHO	displays information about all connected users	2/3

(cont.)

COMMAND	DESCRIPTION	Part/Chapter
WRTALL	sends a message to all users	2/3
WRTUSER	sends a message to a specific user	2/3
WSTAT	displays the PDD physical description of a directory	2/3

SHELL COMMANDS FUNCTION PERMUTED INDEX

displays printer and spooling class : (job queue) status.....LPQ(2/3)
converts a file from a 6.0 to a : 5.2 file format.....TAMROF(2/3)
converts a file from a : 5.2 to a 6.0 file format.....FORMAT(2/3)
converts a file from a 5.2 to a : 6.0 file format.....FORMAT(2/3)
converts a file from a : 6.0 to a 5.2 file format.....TAMROF(2/3)
converts a packed positional file for release : 6.0.....CONVPACK(2/3)
: aborts a family.....KILLFAM(3/4)*
: aborts a running user job.....KILL(2/3)
changes the : access rights of a file system object....SETMODE(2/3)
reads or modifies the default : access rights of new files.....RIGHTS(**)
creates an : alias file.....MKALIAS(2/3)
displays the contents of an : alias file.....SHALIAS(2/3)
displays information about : all connected users.....WHO(2/3)
decreases the priority of : all subsequent programs.....PRIOR(**)
sends a message to : all users.....WRTALL(2/3)
creates a volume or : allocates space for a memory dump.....MKVOL(2/3)
MOS system/ : allows file transfer from S6000 to L1....FFT(2/3)
spooled jobs/ : allows normal user control of own.....SPOOL(2/3)
jobs/ : allows normal user some control of own...BATCH(2/3)
users and groups/ : allows normal user to list registered....USERMAN(2/3)
environment/ : allows sys. admin. control of batch.....BATCH(3/4)*
environment/ : allows sys. admin. control of spooling...SPOOL(3/4)*
users and groups/ : allows sys. admin. to manage system.....USERMAN(3/4)*
: assigns private buffers to a file.....SETBUF(2/3)
modifies the : attributes of a given file.....SETINFO(2/3)
displays the : batch classes.....BMQ(2/3)
allows sys. admin. control of : batch environment.....BATCH(3/4)*
executes a command in : batch.....BM(**)
shares printer or cash adapter : between 2 workstations.....DEVSHARE(3/4)*
releases a connection : between system and volume.....UNMNT(2/3)
assigns private : buffers to a file.....SETBUF(2/3)
displays a : byte-stream file page by page.....MORE(2/3)
creates a : byte-stream file.....MKBST(2/3)
packs a : byte-stream file.....PACK(2/3)
unpacks a : byte-stream file.....UNPACK(2/3)
: calculates the length of a given string..LENGTH(**)
shares printer or : cash adapter between 2 workstations.....DEVSHARE(3/4)*
manages the : centralised system messages database.....MKCEM(3/4)*
system object/ : changes the access rights of a file.....SETMODE(2/3)
or volume/ : changes the name of a file, directory,...RENAME(2/3)
or volume/ : changes the owner of a file, directory,..SETIDEN(3/4)*
displays the current terminal : characteristics.....SHTERM(2/3)
: checks if a file exists.....EXIST(2/3)
: checks the syntax of an MCL procedure....EDPARSE(**)
displays printer and spooling : class (job queue) status.....LPQ(2/3)
connects a local name to a SPOOL : class.....CONSP(**)
displays the batch : classes.....BMQ(2/3)
: clears the whole screen.....CLEAR(2/3)
executes a : command in batch.....BM(**)
displays the Shell : commands help file.....HELP(2/3)
: compacts a disc file.....COMPACT(2/3)
: compacts a keyed file.....CMPK(2/3)
: compacts a volume.....VCOMPACT(3/4)*
: compares files or devices.....DIFF(2/3)
displays information about all : connected users.....WHO(2/3)
displays the local and global names : connected.....SHCON(**)
releases a : connection between system and volume.....UNMNT(2/3)
: connects a local name to a global name...CONN(**)
: connects a local name to a SPOOL class...CONSP(**)
to the system/ : connects a physical volume on a drive....MNT(2/3)
deletes the : contents of a directory.....CLEARDIR(2/3)
list the : contents of a directory.....SHDIR(2/3)

displays the : contents of a given file.....LIST(2/3)
displays the : contents of an alias file.....SHALIAS(2/3)
allows sys. admin. : control of batch environment.....BATCH(3/4)*
allows normal user some : control of own jobs.....BATCH(2/3)
allows normal user : control of own spooled jobs.....SPOOL(2/3)
allows sys. admin. : control of spooling environment.....SPOOL(3/4)*
performs standard data : conversion on floppy disc.....FCU(2/3)
performs standard data : conversion on MTU.....TCU(2/3)
creates a user : conversion table for use by FCU.....FCUTAB(2/3)
performs directory-program directory : conversion.....CHTYPE(2/3)
file format/ : converts a file from a 5.2 to a 6.0.....FORMAT(2/3)
file format/ : converts a file from a 6.0 to a 5.2.....TAMROF(2/3)
release 6.0/ : converts a packed positional file for....CONVPACK(2/3)
: converts the type of a file.....CONVERT(2/3)
recursive mode/ : copies a directory or volume in.....CPTREE(2/3)
: copies a file, directory, or volume.....COPY(2/3)
: creates a byte-stream file.....MKBST(2/3)
: creates a directory.....MKDIR(2/3)
: creates a keyed file.....MKKEYED(2/3)
: creates a positional file.....MKPOS(2/3)
file/ : creates a secondary index in a keyed.....MKINDEX(2/3)
procedure/ : creates a trace file for an MCL.....TRACE(**)
by FCU/ : creates a user conversion table for use..FCUTAB(2/3)
a memory dump/ : creates a volume or allocates space for..MKVOL(2/3)
: creates an alias file.....MKALIAS(2/3)
: creates or modifies a password.....SETPASSW(2/3)
displays the : current terminal characteristics.....SHTERM(2/3)
performs standard : data conversion on floppy disc.....FCU(2/3)
performs standard : data conversion on MTU.....TCU(2/3)
manages the centralised system messages : database.....MKCEM(3/4)*
sets or modifies the system : date and time.....SETDATE(3/4)*
displays the system : date and time.....SHDATE(2/3)
subsequent programs/ : decreases the priority of all.....PRIOR(**)
reads or modifies the : default access rights of new files.....RIGHTS(**)
: deletes an index from a keyed file.....DELINDEX(2/3)
: deletes files, directories, or volumes...REMOVE(2/3)
: deletes the contents of a directory.....CLEARDIR(2/3)
displays the PDD physical : description of a directory.....WSTAT(2/3)
displays the PDD physical : description of a file.....RP(2/3)
displays the : descriptor of a main volume.....SHLABEL(2/3)
compares files or : devices.....DIFF(2/3)
deletes files, : directories, or volumes.....REMOVE(2/3)
performs directory-program : directory conversion.....CHTYPE(2/3)
copies a : directory or volume in recursive mode....CPTREE(2/3)
dumps a file, : directory, or volume onto floppy disc....FLD(2/3)
dumps a file, : directory, or volume onto MTU.....FILETAR(2/3)
copies a file, : directory, or volume.....COPY(2/3)
displays information on a file, : directory, or volume.....LS(2/3)
displays information on a file, : directory, or volume.....PRY(2/3)
changes the name of a file, : directory, or volume.....RENAME(2/3)
changes the owner of a file, : directory, or volume.....SETIDEN(3/4)*
searches for a given file, : directory, or volume.....WHEREIS(2/3)
performs : directory-program directory conversion...CHTYPE(2/3)
deletes the contents of a : directory.....CLEARDIR(2/3)
creates a : directory.....MKDIR(2/3)
sets the working : directory.....SETWDIR(**)
list the contents of a : directory.....SHDIR(2/3)
displays the path name of the working : directory.....SHWDIR(**)
displays the PDD physical description of a : directory.....WSTAT(2/3)
compacts a : disc file.....COMPACT(2/3)
performs standard data conversion on floppy : disc.....FCU(2/3)
dumps a file, directory, or volume onto floppy : disc.....FLD(2/3)

initialises track 0 of a floppy : disc.....MKENV(3/4)*
: disconnects a local name.....DISCON(**)
page/ : displays a byte-stream file page by.....MORE(2/3)
: displays family status information.....SHFAM(2/3)
connected users/ : displays information about all.....WHO(2/3)
directory, or volume/ : displays information on a file,.....LS(2/3)
directory, or volume/ : displays information on a file,.....PRY(2/3)
(job queue) status/ : displays printer and spooling class.....LPQ(2/3)
: displays software release information....RCODE(2/3)
: displays the batch classes.....BMQ(2/3)
: displays the contents of a given file....LIST(2/3)
: displays the contents of an alias file...SHALIAS(2/3)
characteristics/ : displays the current terminal.....SHTERM(2/3)
volume/ : displays the descriptor of a main.....SHLABEL(2/3)
its size/ : displays the History List or modifies....LASTCOM(**)
connected/ : displays the local and global names.....SHCON(**)
: displays the machine name.....MHNAME(2/3)
directory/ : displays the path name of the working....SHWDIR(**)
of a file/ : displays the PDD physical description....RP(2/3)
of a directory/ : displays the PDD physical description....WSTAT(2/3)
: displays the result of an expression.....ECHO(**)
: displays the result of an expression.....SHOW(2/3)
: displays the Shell commands help file....HELP(2/3)
: displays the system date and time.....SHDATE(2/3)
: displays the user's login name.....SHNAME(**)
: displays the value of an MCL variable....SHVAR(**)
: displays user information.....FINGER(2/3)
connects a physical volume on a : drive to the system.....MNT(2/3)
creates a volume or allocates space for a memory : dump.....MKVOL(2/3)
MTU/ : dumps a file, directory, or volume onto..FILETAR(2/3)
floppy disc/ : dumps a file, directory, or volume onto..FLD(2/3)
: dumps a volume onto SCT.....VOLSR(2/3)
allows sys. admin. control of batch : environment.....BATCH(3/4)*
exits from the Shell : environment.....LOGOUT(**)
allows sys. admin. control of spooling : environment.....SPOOL(3/4)*
: executes a command in batch.....BM(**)
suspends the job in : execution.....SUSPEND(2/3)
checks if a file : exists.....EXIST(2/3)
: exits from the Shell environment.....LOGOUT(**)
displays the result of an : expression.....ECHO(**)
displays the result of an : expression.....SHOW(2/3)
displays : family status information.....SHFAM(2/3)
aborts a : family.....KILLFAM(3/4)*
creates a user conversion table for use by : FCU.....FCUTAB(2/3)
checks if a : file exists.....EXIST(2/3)
creates a trace : file for an MCL procedure.....TRACE(**)
converts a packed positional : file for release 6.0.....CONVPACK(2/3)
converts a file from a 5.2 to a 6.0 : file format.....FORMAT(2/3)
converts a file from a 6.0 to a 5.2 : file format.....TAMROF(2/3)
converts a : file from a 5.2 to a 6.0 file format.....FORMAT(2/3)
converts a : file from a 6.0 to a 5.2 file format.....TAMROF(2/3)
displays a byte-stream : file page by page.....MORE(2/3)
changes the access rights of a : file system object.....SETMODE(2/3)
system/ allows : file transfer from S6000 to L1 MOS.....FFT(2/3)
disc/ dumps a : file, directory, or volume onto floppy...FLD(2/3)
dumps a : file, directory, or volume onto MTU.....FILETAR(2/3)
copies a : file, directory, or volume.....COPY(2/3)
displays information on a : file, directory, or volume.....LS(2/3)
displays information on a : file, directory, or volume.....PRY(2/3)
changes the name of a : file, directory, or volume.....RENAME(2/3)
changes the owner of a : file, directory, or volume.....SETIDEN(3/4)*
searches for a given : file, directory, or volume.....WHEREIS(2/3)

compacts a keyed : file.....CMPK(2/3)
compacts a disc : file.....COMPACT(2/3)
converts the type of a : file.....CONVERT(2/3)
deletes an index from a keyed : file.....DELINDEX(2/3)
displays the Shell commands help : file.....HELP(2/3)
displays the contents of a given : file.....LIST(2/3)
creates an alias : file.....MKALIAS(2/3)
creates a byte-stream : file.....MKBST(2/3)
creates a secondary index in a keyed : file.....MKINDEX(2/3)
creates a keyed : file.....MKKEYED(2/3)
creates a positional : file.....MKPOS(2/3)
packs a byte-stream : file.....PACK(2/3)
displays the PDD physical description of a : file.....RP(2/3)
assigns private buffers to a : file.....SETBUF(2/3)
modifies the attributes of a given : file.....SETINFO(2/3)
displays the contents of an alias : file.....SHALIAS(2/3)
unpacks a byte-stream : file.....UNPACK(2/3)
compares : files or devices.....DIFF(2/3)
deletes : files, directories, or volumes.....REMOVE(2/3)
prints : files.....LPR(2/3)
reads or modifies the default access rights of new : files.....RIGHTS(**)
performs standard data conversion on : floppy disc.....FCU(2/3)
dumps a file, directory, or volume onto : floppy disc.....FLD(2/3)
initialises track 0 of a : floppy disc.....MKENV(3/4)*
converts a file from a 5.2 to a 6.0 file : format.....FORMAT(2/3)
converts a file from a 6.0 to a 5.2 file : format.....TAMROF(2/3)
connects a local name to a : global name.....CONN(**)
displays the local and : global names connected.....SHCON(**)
allows normal user to list registered users and : groups.....USERMAN(2/3)
allows sys. admin. to manage system users and : groups.....USERMAN(3/4)*
displays the Shell commands : help file.....HELP(2/3)
displays the : History List or modifies its size.....LASTCOM(**)
deletes an : index from a keyed file.....DELINDEX(2/3)
creates a secondary : index in a keyed file.....MKINDEX(2/3)
displays : information about all connected users....WHO(2/3)
volume/ displays : information on a file, directory, or.....LS(2/3)
volume/ displays : information on a file, directory, or.....PRY(2/3)
displays user : information.....FINGER(2/3)
displays software release : information.....RCODE(2/3)
displays family status : information.....SHFAM(2/3)
: initialises track 0 of a floppy disc.....MKENV(3/4)*
: installs or uninstalls a MOS package.....INSTALPKG(3/4)*
suspends the : job in execution.....SUSPEND(2/3)
aborts a running user : job.....KILL(2/3)
allows normal user some control of own : jobs.....BATCH(2/3)
allows normal user control of own spooled : jobs.....SPOOL(2/3)
compacts a : keyed file.....CMPK(2/3)
deletes an index from a : keyed file.....DELINDEX(2/3)
creates a secondary index in a : keyed file.....MKINDEX(2/3)
creates a : keyed file.....MKKEYED(2/3)
calculates the : length of a given string.....LENGTH(**)
allows normal user to : list registered users and groups.....USERMAN(2/3)
: list the contents of a directory.....SHDIR(2/3)
displays the : local and global names connected.....SHCON(**)
connects a : local name to a global name.....CONN(**)
connects a : local name to a SPOOL class.....CONSP(**)
disconnects a : local name.....DISCON(**)
displays the user's : login name.....SHNAME(**)
displays the : machine name.....MHNAME(2/3)
displays the descriptor of a : main volume.....SHLABEL(2/3)
allows sys. admin. to : manage system users and groups.....USERMAN(3/4)*
database/ : manages the centralised system messages..MKCEM(3/4)*

sends a request to the : Master WorkStation.....CALLMWS(2/3)
reads the messages sent to the : Master WorkStation.....MTA(3/4)*
checks the syntax of an : MCL procedure.....EDPARSE(**)
creates a trace file for an : MCL procedure.....TRACE(**)
displays the value of an : MCL variable.....SHVAR(**)
creates a volume or allocates space for a : memory dump.....MKVOL(2/3)
sends a : message to a specific user.....WRTUSER(2/3)
sends a : message to all users.....WRTALL(2/3)
manages the centralised system : messages database.....MKCEM(3/4)*
reads the : messages sent to the Master WorkStation..MTA(3/4)*
copies a directory or volume in recursive : mode.....CPTREE(2/3)
creates or : modifies a password.....SETPASSW(2/3)
displays the History List or : modifies its size.....LASTCOM(**)
: modifies the attributes of a given file..SETINFO(2/3)
new files/ reads or : modifies the default access rights of....RIGHTS(**)
sets or : modifies the system date and time.....SETDATE(3/4)*
installs or uninstalls a : MOS package.....INSTALPKG(3/4)*
allows file transfer from S6000 to L1 : MOS system.....FFT(2/3)
dumps a file, directory, or volume onto : MTU.....FILETAR(2/3)
performs standard data conversion on : MTU.....TCU(2/3)
changes the : name of a file, directory, or volume.....RENAME(2/3)
displays the path : name of the working directory.....SHWDIR(**)
connects a local : name to a global name.....CONN(**)
connects a local : name to a SPOOL class.....CONSP(**)
connects a local name to a global : name.....CONN(**)
disconnects a local : name.....DISCON(**)
displays the machine : name.....MHNAME(2/3)
displays the user's login : name.....SHNAME(**)
displays the local and global : names connected.....SHCON(**)
reads or modifies the default access rights of : new files.....RIGHTS(**)
allows : normal user control of own spooled jobs..SPOOL(2/3)
allows : normal user some control of own jobs.....BATCH(2/3)
and groups/ allows : normal user to list registered users.....USERMAN(2/3)
changes the access rights of a file system : object.....SETMODE(2/3)
allows normal user some control of : own jobs.....BATCH(2/3)
allows normal user control of : own spooled jobs.....SPOOL(2/3)
changes the : owner of a file, directory, or volume....SETIDEN(3/4)*
installs or uninstalls a MOS : package.....INSTALPKG(3/4)*
converts a : packed positional file for release 6.0...CONVPACK(2/3)
: packs a byte-stream file.....PACK(2/3)
creates or modifies a : password.....SETPASSW(2/3)
displays the : path name of the working directory.....SHWDIR(**)
displays the : PDD physical description of a directory..WSTAT(2/3)
displays the : PDD physical description of a file.....RP(2/3)
conversion/ : performs directory-program directory.....CHTYPE(2/3)
floppy disc/ : performs standard data conversion on.....FCU(2/3)
MTU/ : performs standard data conversion on.....TCU(2/3)
displays the PDD : physical description of a directory.....WSTAT(2/3)
displays the PDD : physical description of a file.....RP(2/3)
system/ connects a : physical volume on a drive to the.....MNT(2/3)
converts a packed : positional file for release 6.0.....CONVPACK(2/3)
creates a : positional file.....MKPOS(2/3)
status/ displays : printer and spooling class (job queue)...LPQ(2/3)
workstations/ shares : printer or cash adapter between 2.....DEVSHARE(3/4)*
: prints files.....LPR(2/3)
decreases the : priority of all subsequent programs.....PRIOR(**)
assigns : private buffers to a file.....SETBUF(2/3)
checks the syntax of an MCL : procedure.....EDPARSE(**)
creates a trace file for an MCL : procedure.....TRACE(**)
resumes a suspended user : program.....RESUME(2/3)
decreases the priority of all subsequent : programs.....PRIOR(**)
displays printer and spooling class (job : queue) status.....LPQ(2/3)

rights of new files/ : reads or modifies the default access.....RIGHTS(**)
WorkStation/ : reads the messages sent to the Master....MTA(3/4)*
copies a directory or volume in : recursive mode.....CPTREE(2/3)
allows normal user to list : registered users and groups.....USERMAN(2/3)
converts a packed positional file for : release 6.0.....CONVPACK(2/3)
displays software : release information.....RCODE(2/3)
and volume/ : releases a connection between system.....UNMNT(2/3)
sends a : request to the Master WorkStation.....CALLMWS(2/3)
displays the : result of an expression.....ECHO(**)
displays the : result of an expression.....SHOW(2/3)
: resumes a suspended user program.....RESUME(2/3)
changes the access : rights of a file system object.....SETMODE(2/3)
reads or modifies the default access : rights of new files.....RIGHTS(**)
aborts a : running user job.....KILL(2/3)
allows file transfer from : S6000 to L1 MOS system.....FFT(2/3)
sets the : screen size.....SETTERM(2/3)
clears the whole : screen.....CLEAR(2/3)
dumps a volume onto : SCT.....VOLSR(2/3)
or volume/ : searches for a given file, directory,....WHEREIS(2/3)
creates a : secondary index in a keyed file.....MKINDEX(2/3)
: sends a message to a specific user.....WRTUSER(2/3)
: sends a message to all users.....WRTALL(2/3)
WorkStation/ : sends a request to the Master.....CALLMWS(2/3)
reads the messages : sent to the Master WorkStation.....MTA(3/4)*
time/ : sets or modifies the system date and.....SETDATE(3/4)*
: sets the screen size.....SETTERM(2/3)
: sets the working directory.....SETWDIR(**)
2 workstations/ : shares printer or cash adapter between...DEVSHARE(3/4)*
displays the : Shell commands help file.....HELP(2/3)
exits from the : Shell environment.....LOGOUT(**)
displays the History List or modifies its : size.....LASTCOM(**)
sets the screen : size.....SETTERM(2/3)
displays : software release information.....RCODE(2/3)
creates a volume or allocates : space for a memory dump.....MKVOL(2/3)
sends a message to a : specific user.....WRTUSER(2/3)
connects a local name to a : SPOOL class.....CONSP(**)
allows normal user control of own : spooled jobs.....SPOOL(2/3)
displays printer and : spooling class (job queue) status.....LPQ(2/3)
allows sys. admin. control of : spooling environment.....SPOOL(3/4)*
performs : standard data conversion on floppy disc..FCU(2/3)
performs : standard data conversion on MTU.....TCU(2/3)
displays family : status information.....SHFAM(2/3)
displays printer and spooling class (job queue) : status.....LPQ(2/3)
calculates the length of a given : string.....LENGTH(**)
decreases the priority of all : subsequent programs.....PRIOR(**)
resumes a : suspended user program.....RESUME(2/3)
: suspends the job in execution.....SUSPEND(2/3)
checks the : syntax of an MCL procedure.....EDPARSE(**)
environment/ allows : sys. admin. control of batch.....BATCH(3/4)*
environment/ allows : sys. admin. control of spooling.....SPOOL(3/4)*
groups/ allows : sys. admin. to manage system users and...USERMAN(3/4)*
releases a connection between : system and volume.....UNMNT(2/3)
sets or modifies the : system date and time.....SETDATE(3/4)*
displays the : system date and time.....SHDATE(2/3)
manages the centralised : system messages database.....MKCEM(3/4)*
changes the access rights of a file : system object.....SETMODE(2/3)
allows sys. admin. to manage : system users and groups.....USERMAN(3/4)*
allows file transfer from S6000 to L1 MOS : system.....FFT(2/3)
connects a physical volume on a drive to the : system.....MNT(2/3)
creates a user conversion : table for use by FCU.....FCUTAB(2/3)
displays the current : terminal characteristics.....SHTERM(2/3)
sets or modifies the system date and : time.....SETDATE(3/4)*

displays the system date and	time	SHDATE(2/3)
creates a	trace file for an MCL procedure	TRACE(**)
initialises	track 0 of a floppy disc	MKENV(3/4)*
allows file	transfer from S6000 to L1 MOS system	FFT(2/3)
converts the	type of a file	CONVERT(2/3)
installs or	uninstalls a MOS package	INSTALPKG(3/4)*
	unpacks a byte-stream file	UNPACK(2/3)
creates a	user conversion table for use by FCU	FCUTAB(2/3)
displays	user information	FINGER(2/3)
aborts a running	user job	KILL(2/3)
resumes a suspended	user program	RESUME(2/3)
displays the	user's login name	SHNAME(**)
sends a message to a specific	user	WRTUSER(2/3)
allows normal user to list registered	users and groups	USERMAN(2/3)
allows sys. admin. to manage system	users and groups	USERMAN(3/4)*
displays information about all connected	users	WHO(2/3)
sends a message to all	users	WRTALL(2/3)
displays the	value of an MCL variable	SHVAR(**)
displays the value of an MCL	variable	SHVAR(**)
copies a directory or	volume in recursive mode	CPTREE(2/3)
connects a physical	volume on a drive to the system	MNT(2/3)
dumps a file, directory, or	volume onto floppy disc	FLD(2/3)
dumps a file, directory, or	volume onto MTU	FILETAR(2/3)
dumps a	volume onto SCT	VOLSR(2/3)
dump/ creates a	volume or allocates space for a memory	MKVOL(2/3)
copies a file, directory, or	volume	COPY(2/3)
displays information on a file, directory, or	volume	LS(2/3)
displays information on a file, directory, or	volume	PRY(2/3)
changes the name of a file, directory, or	volume	RENAME(2/3)
changes the owner of a file, directory, or	volume	SETIDEN(3/4)*
displays the descriptor of a main	volume	SHLABEL(2/3)
releases a connection between system and	volume	UNMNT(2/3)
compacts a	volume	VCOMPACT(3/4)*
searches for a given file, directory, or	volume	WHEREIS(2/3)
deletes files, directories, or	volumes	REMOVE(2/3)
clears the	whole screen	CLEAR(2/3)
sets the	working directory	SETWDIR(**)
displays the path name of the	working directory	SHWDIR(**)
sends a request to the Master	WorkStation	CALLMWS(2/3)
reads the messages sent to the Master	WorkStation	MTA(3/4)*
shares printer or cash adapter between 2	workstations	DEVSHARE(3/4)*

PART 1 - THE SHELL ENVIRONMENT

INTRODUCTION TO PART 1

Part 1 provides all the information necessary to start working in the Shell environment.

Chapter 1 ("General Characteristics") describes the video and keyboard, the master workstation, how to load and execute commands, the general characteristics of the files .INIT and .MESG, and the operating procedures.

Chapter 2 ("SPOOL System and Batch Management") describes the SPOOL environment and the batch monitor.

1. GENERAL CHARACTERISTICS

THE VIDEO

Videos are available with 13, 16, or 25 lines. At configuration time, their layout can be arranged either to:

- use all the screen as a single area, or to
- divide the screen into two "windows", a large one in the upper part for the user video, and a small one below for the "system line".

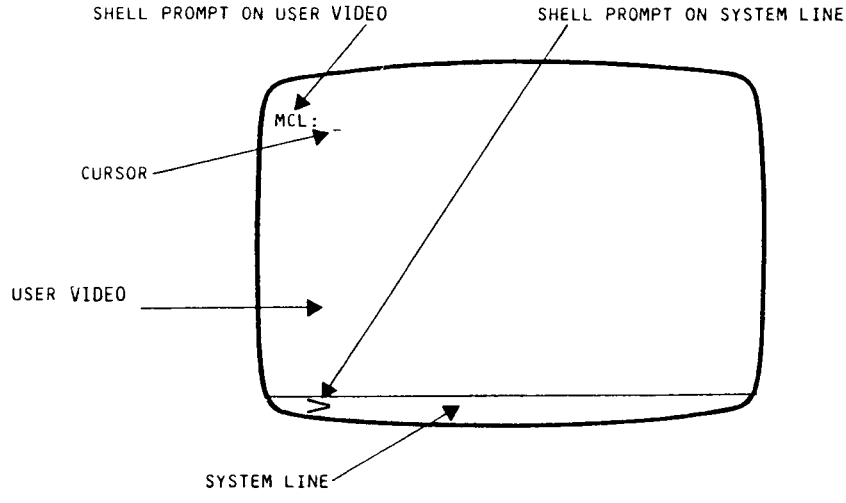


Fig. 1-1 The Shell Video divided into two Windows

USER VIDEO

This is the part in which normal user activities take place: entering commands, activating procedures, or any other Shell activities.

This user video will be referred to simply as "the video". Its default prompt is "MCL:".

SYSTEM LINE

This is the bottom line of the screen. It is activated by pressing **CONTROL K** , which displays the system line prompt (">").

This line is completely independent of the remainder of the screen, and can be used without waiting for other operations to end, (that is, asynchronously), for:

- activating jobs (commands, programs, or procedures)
- controlling jobs being executed
- the immediate display of messages from the operating system
- displaying messages produced by a WRITE command in an MCL procedure being executed (messages longer than 77 characters are stored in the file .MESG).

FULL SCREEN VIDEO

Its default prompt is also "MCL:".

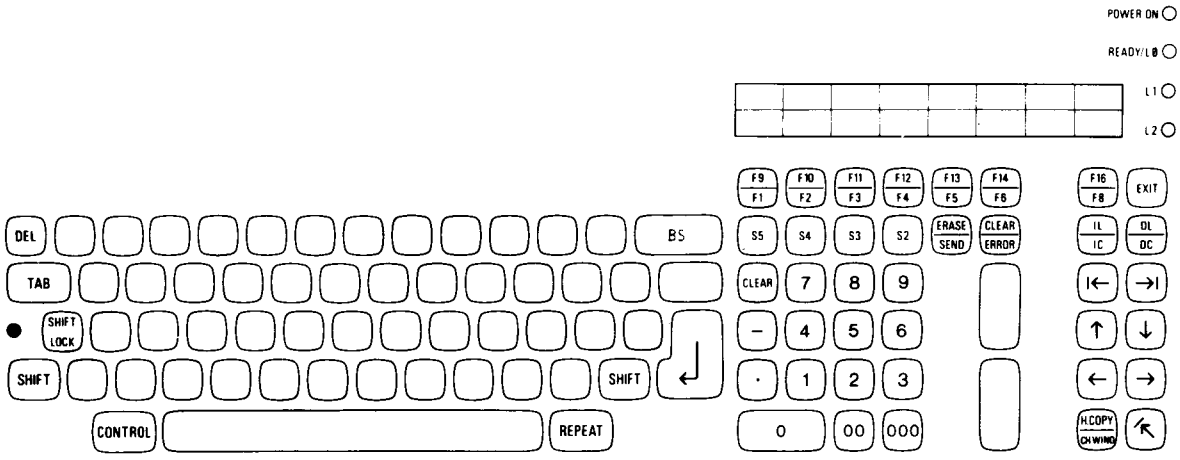
Pressing **CONTROL K** in full screen video mode kills the process being executed.

The commands SUSPEND and RESUME are only available if the system line is present on the video.

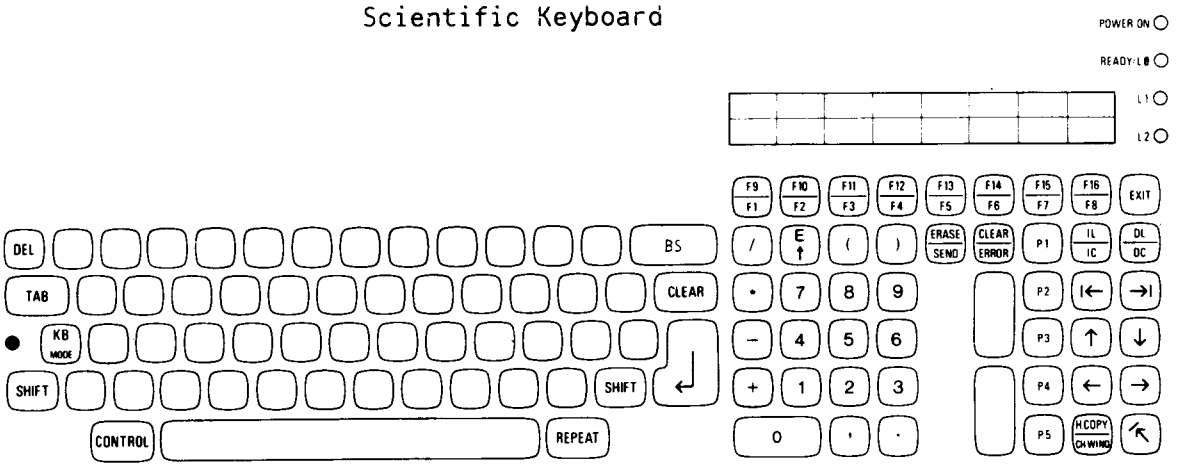
THE KEYBOARD

The various keyboard layouts (Scientific, Data Processing, and Multifunctional) are illustrated here.

Data Processing Keyboard



Scientific Keyboard



Multifunctional Keyboard

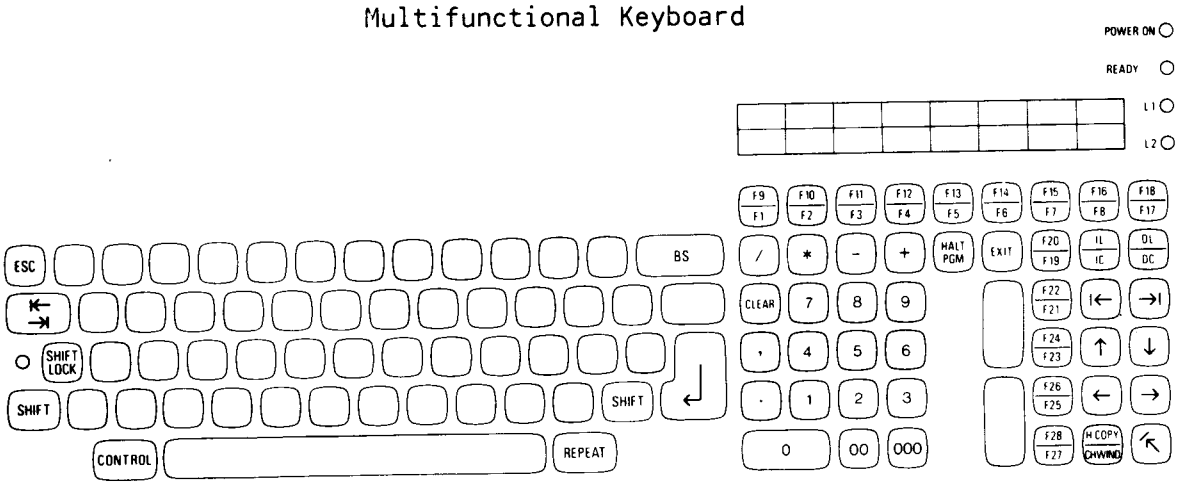


Fig. 1-2 Keyboard Layouts

Few function keys perform an action in the Shell environment. Function keys which have no Shell function may display codes but have no other effect.

Valid function keys are:

CH.WIN shifts the cursor from the video to the system line and vice versa.

CONTROL K behaves as configured: it may activate the system line, or kill a process in execution.

CLEAR cancels all the characters entered on the current line except the prompt (Data Processing keyboards only).

***** cancels all the characters entered on the current line except the prompt (Scientific keyboard only).

CONTROL S stops the output display on the video.

CONTROL Q resumes the output display on the video.

A useful keyboard feature allows the user to continue entering other commands while one command is executing. These are stored in a 160 characters buffer. An audible "beep" warns of a full buffer, and further characters are not stored until the contents of the buffer have been read. Commands entered thus are executed as soon as the previous command is completed.

MASTER WORKSTATION

The master workstation (MWS) consists of a warning terminal and a master terminal, and is defined in Grandpa during initialisation. It receives calls from all users, collecting messages, data, and requests for operator intervention.

WARNING TERMINAL

This is the bottom line of a video which has been specifically configured. It must always be available for asynchronous display of warning messages (maximum of 80 characters: 30 being reserved for the system and the rest for the user).

MASTER TERMINAL

This consists of the first 12, 15, or 24 lines of a video on which the MTA program is active (not necessarily the one containing the warning terminal).

It handles messages from users which the system has put in the MWS queue.

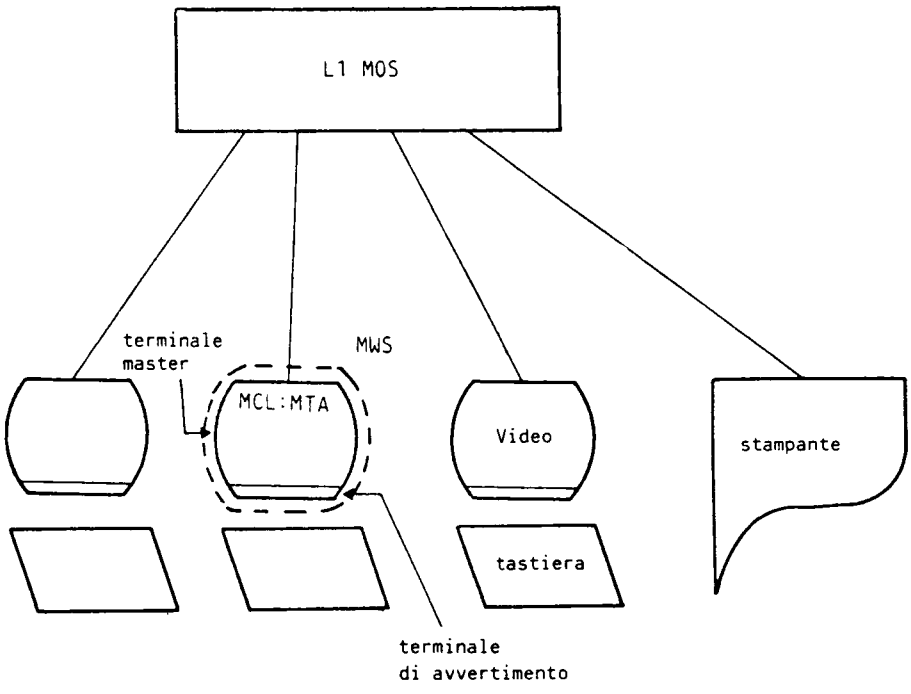


Fig. 1-3 Example of a Master Workstation

TYPES OF MESSAGES

Users can send messages of the following type to the MWS:

- warning message
- data
- data/information with wait for a reply.

Warning messages are displayed asynchronously on the warning terminal. Data and information are enqueued, and displayed on the terminal which requests them with the MTA command (master terminal).

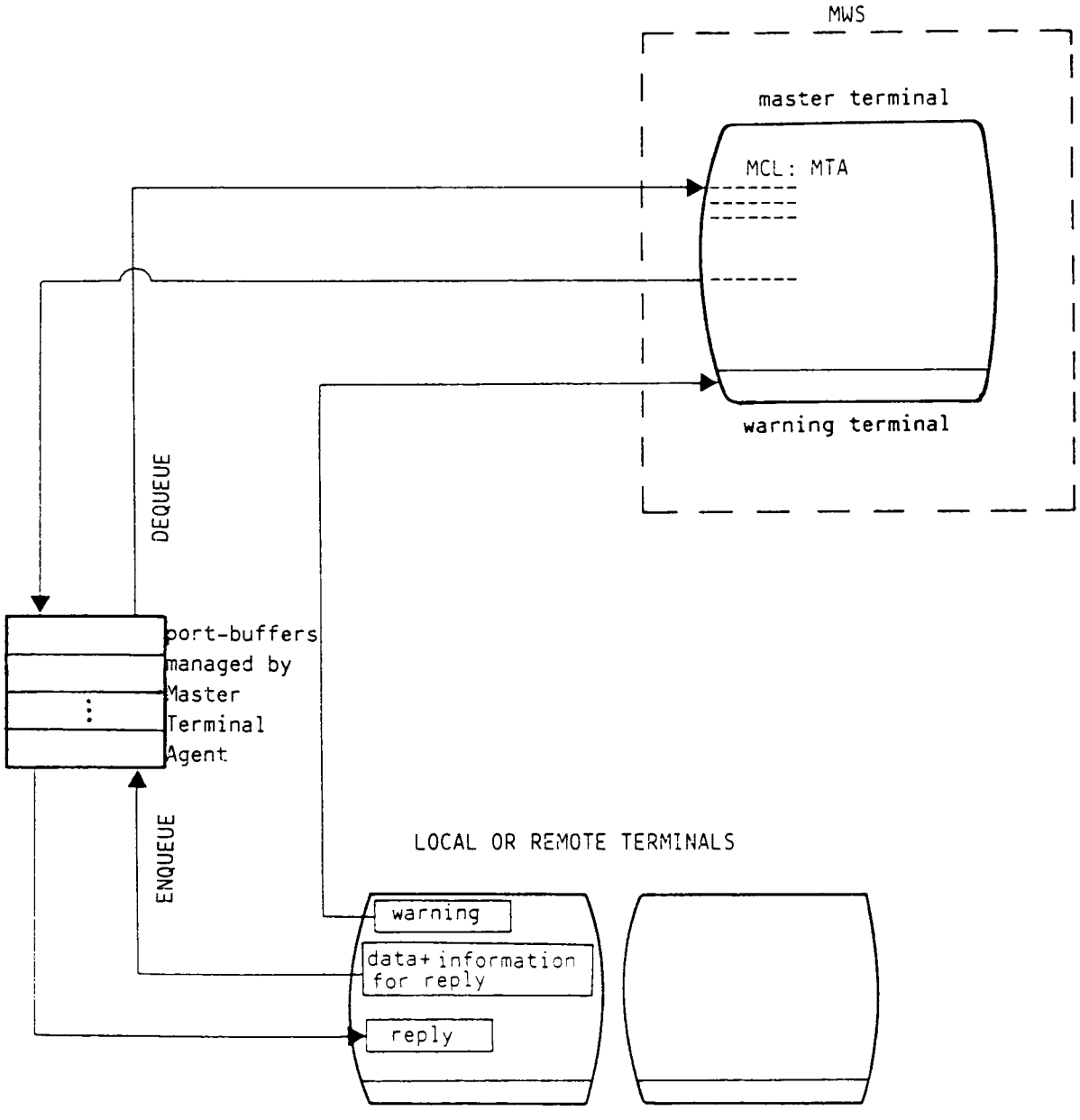


Fig. 1-4 Effects of Operator Intervention Requests

LOCAL AND GLOBAL MWS

There is always a master workstation in the system.

In a cluster configuration there can be several local master workstations, one for each satellite.

The local MWS is only accessible to the users that are local to the satellite.

The global MWS of the master system can be accessed by all the system users.

In a local network there is a local MWS on each machine. A global MWS can be defined, which will be the only one in the network.

Note: If the global MWS of a local network which includes a cluster does not belong to the master system of the cluster, then it is not available to the satellites of that cluster.

LOADING AND EXECUTING COMMANDS

Commands may be input and executed individually, or as a group forming a procedure. Such a group must be entered as an MCL language input line, being a series of MCL commands, procedures and/or program names, each separated by semicolons (;). The maximum number of characters per line is specified in Appendix B.

Each command consists of a keyword expressing the function of the command, followed by one or more parameters (see below).

VIDEO INPUT LINE

A video input line can be up to 80 characters long, the first 5 characters being the Shell prompt "MCL: " (4 characters and a blank), so that the user can enter 75 characters. After position 80 in the first line, the cursor shifts automatically to the next line, where another 80 characters can be entered. On reaching the 155th character, the cursor again shifts to the next line, where 5 characters can be entered (the 5th character must be a carriage return).

SYSTEM LINE INPUT

The system line is similar to the video input line, with two differences: firstly, there are only 2 characters in the prompt, (">" and a blank); secondly, after the 78th or 158th character has been keyed (the final character must be carriage return), the system line is stored and erased from the screen.

A command is entered on the system line by:

- pressing the **CH.WIN** key to shift the cursor to the system line
- pressing **CONTROL K** to activate the system line
- keying the command string.

After the requested activity has been executed, the cursor automatically reappears on the video.

Note: It is always advisable to provide a system line: otherwise, commands which normally send messages to it cannot do so, and the user must operate "blindly".

COMMAND PARAMETERS

Command parameters are positional and are separated by one or more blanks. Some commands can also have keyed parameters. The parameter can be a string of characters, or an expression which is evaluated as the command gets executed.

Parameter Examples

Examples of commands and their parameters follow:

MKPOS /IPL/USR/JOHN/KENT RCDLG=128 "D	(positional and keyed parameters)
CONVERT LONDON P K START=50 KEYLG=53	(positional and keyed parameters)
MKKEYED %NAME&%A1 RCDLG=1024 START=31 KEYLG=26	(positional parameter defined by expression and keyed parameters)

OPTIONS

Many commands allow their functions to be selected by options. Such options are indicated by " (double quotes) and a letter or string, so that the function may be selected by:

"R or "a or "L or "VR or "file name .

Some options may be specified using either upper or lower case letters: upper case (capital letters) is always accepted. (Command descriptions always show options in upper case, for clarity.)

INTERACTIVE COMMANDS

Many MCL commands will accept parameters supplied interactively: first the user enters the command on its own, then each parameter value is requested in turn by an appropriate prompt displayed on the screen.

ASYNCHRONOUS CONTROL COMMANDS

The Shell commands SUSPEND, RESUME, and KILL affect jobs in execution without waiting for their completion (that is, asynchronously).

SUSPEND holds up a job for as long as a user requires, perhaps to allow other jobs to run.

RESUME allows a suspended job to continue.

KILL aborts a job in execution.

Note: KILL and SUSPEND must be entered on the system line. When one of these has been executed, the Shell prompt reappears on the screen and the user can operate normally.

The RESUME command must be entered on the video input line.

MCL COMMAND EXECUTION IN ASYNCHRONOUS MODE

If a user needs to execute a single job urgently, he can do so even before execution of the current procedure or program is completed (that is, in asynchronous mode).

This is done by entering the required command for the single job on the system line. This suspends the activity in process (without the user having to specify SUSPEND), executes the command on the system line, then automatically resumes execution of the suspended activity (without the use of RESUME).

BUILT-IN COMMANDS

These are integral to the Shell interpreter, and are therefore available wherever it is used, whatever its configuration. They are described in the manual MCL MOS Command Language User Guide, and are:

BM	LENGTH	SHNAME
CONN	PRIOR	SHVAR
CONSP	RDNEXT	SHWDIR
DISCON	RDPOS	SUBSTR
ECHO	READ	TRACE
EDPARSE	RIGHTS	WRITE
LASTCOM	SETWDIR	
LOGOUT	SHCON	

REMOTE COMMANDS

A command which operates on remote objects (files, directories, etc.) is known as a remote command (see Appendix B for the complete list.)

The path name specified in the commands must be preceded by:

/../machine-name/

where "machine-name" is NxMy (Nx = network identifier, My = machine identifier), see MOS Distributed System in Local Area Network (LAN) User Guide.

COMPLETION CODE

Whether Shell commands are activated interactively or from within an MCL procedure, on completion of the operation (normal or abnormal), they return a code which is stored by the system. The code informs the user of the outcome of the operation. It is contained by the MCL variable %STATUS. For completion code interpretation, see the manual MCL MOS Command Language User Guide.

REDIRECTION OF INPUT AND OUTPUT

Commands that receive input from the keyboard or display output on the video can, unless it is explicitly forbidden, be made to receive input from another source, or to write output to another destination. This is achieved by "redirection". See MCL MOS Command Language User Guide.

Redirection is a means of storing data for a longer period than is usual for input or output data. It is particularly useful when data is used repeatedly. For example, when an interactive command is used repeatedly with the same keyboard input, the input may be redirected. If they are to be used in an MCL procedure, such command parameters may be redirected. Similarly, output from one procedure may be redirected, to be read as input by another. In such cases, it is possible to create a file containing the required data, and use redirection so that input is read from that file.

The input file must be created using the system editor. For the parameter values, each line must contain the answer to a command prompt. When the default value is required for a parameter, it is still selected by means of carriage return, and is then represented on the file by a blank line. That blank line in the input is then interpreted as carriage return.

Redirection of output to a file for preservation is especially useful when output data might be lost from a display due to scrolling of the screen.

ERROR HANDLING

In the Shell environment, errors are notified by means of explanatory messages.

Errors in commands entered on the system line are displayed in the video section. An error in a command in the input line automatically cancels subsequent commands in the same line.

There are two basic types of errors in the Shell environment:

- MCL language errors (preceded by the comment "MCL ERROR")
- MCL command errors (preceded by the name of the class).

If an unrecognised string is entered at the keyboard, the message:

xx...x NOT FOUND

is displayed, where "xx...x" is the unrecognised string.

The error messages, their meaning and the corrective action to be taken are listed in Appendix A.

COMMAND REPETITION

Shell commands which are not suspended can be repeated, with or without modification. This applies to any command on the History List.

The History List (HL) retains a specified number of previously input commands, executed or not, and their sequence numbers. They are kept in input order, ending with the latest command. The number of commands to be retained is set by the user, using the built-in command LASTCOM. All commands are numbered sequentially, and their sequence numbers also retained.

The trigger ! , together with the data from the history list, can be used to repeat commands, without entering the entire command again. Thus:

!CR	repeats the last executed command, unchanged.
!str CR	repeats the last executed command with the string "str" appended to it.
^old-string^new-string	refers to the last command, replacing the first occurrence of "old-string" with "new-string". "new-string" may be an empty string, or can replace the command name. (For example, ^MORE^LPR prints the file which was just displayed).

COMMAND EDITING

The line editor is triggered by "!!".

The position of **CR** in line editing mode is crucial to the effect of the command.

CR in the first position executes the previous line. In any other position, it displays the command for possible re-editing.

!! CR displays the last command, and places the cursor at the beginning of the next line.

Other line editor commands are:

space	moves the cursor to the right.
backspace	movs the cursor to the left.
D	deletes the character above the cursor.
Istr	inserts the string "str" at the cursor position.

Note: Any number of characters may be deleted at a time, but only one string may be inserted during each pass of the editor.

When used with history list data:

!!HLNo CR (where "HLNo" is the sequence number of a history list command), repeats that command.

For example, **!!123 CR** repeats command number 123.

!!string CR is a variation on the same theme. This starts a search for the latest entry to match the given string, searching backwards from the last entry. The string does not have to be complete. For example, if the history list is:

```
12 SHWDIR
13 SHDIR
14 COPY A B
15 shdir
```

Entered in position 16, **!!SH** would repeat command number 13, but **!!SHW** would repeat command number 12. (The more recent command number 15 would be passed over, not because it is a wrong entry, but because it is in lower case, so it does not match exactly.)

.MESG FILE

Every user may create a .MESG file in his home directory.

This file may also be created by the system when:

- a WRITE command is executed which has a string longer than 78 characters
- a user issues a WRTUSER command when the receiving user is not logged onto the system.

In such an event, one of the following messages appears on the screen of the receiving user as soon as he logs on:

THERE IS A MESSAGE

or

THERE ARE MESSAGES

LIST or MORE can display any messages in a user's .MESG file. REMOVE can delete the .MESG file.

The .MESG file path name takes the form:

/IPL/USR/user-name/.MESG

where "user-name" is the home directory allocated to the user when created.

The system administrator may change the .MESG path name by means of the reserved command USERMAN.

.INIT FILE

The .INIT file is an MCL procedure file that must be created and edited by the user with the EDPARSE command or with the system editor.

The .INIT file (if it exists) is automatically executed immediately after selection of the Shell environment by the user and before the Shell prompt appears. It can contain commands and procedures (separated by semicolons) necessary for a customised initialisation of a Shell work session.

The file path name is as follows:

/IPL/USR/user-name/.INIT

where "/IPL/USR/user-namel" is the home directory allocated to the user when created.

For example, the following .INIT file displays the current software release, the system date and time, and sets the prompt to the first five letters of the user's login name followed by a colon ":".

```
ECHO "L1 MOS REL.6.0" ;
SHDATE ;
SHNAME %NAME ;
SUBSTR %NAME 1 5 %PROMPT ;
SET %PROMPT := %PROMPT & ":" ;
SET %PATH := ". /IPL/CMD /IPL/DCP /IPL/PROGRAMS" ;
```

OPERATING PROCEDURES

LOGIN AND LOGOUT

The user has to "login" before entering the Shell environment from the starting menu. In its simplest form, this consists of keying-in a name previously agreed with the system administrator, which identifies the user to the machine as someone authorised to use its facilities. (It is impossible to login with the same name at more than one workstation).

To exit from the Shell environment and terminate a work session, the user must "logout" by entering the LOGOUT command. This displays the starting menu again, allowing the user either:

- to select another environment
- or to exit from the system completely by selecting LOGOUT.

SYSTEM ADMINISTRATOR FUNCTIONS

"ROOT" is the login name of the system administrator, given when the system was configured.

Only the system administrator can create or delete users, by means of the USERMAN command.

Commands reserved to the system administrator are described in Part 3.

PASSWORD

To prevent unauthorised access on the terminals, each user can provide himself with a password of up to 8 characters, using the SETPASSW command.

2. SPOOL SYSTEM AND BATCH MANAGEMENT

THE SPOOL SYSTEM

Without the SPOOL system, users have access to printers, but each must wait for the printer to be free of other prints before starting his own.

The SPOOL system allows files which need printing to be added to a queue, and printed in turn with the minimum of human intervention.

SPOOL OBJECTS

Objects referred to by SPOOL are classes, printers, unspoolers, and print jobs, whose meaning is as follows:

Class — This is another name for a queue of jobs (files) awaiting printing. A print queue can contain print files produced not only by Shell users, but also office automation users, BEAM, MTS, and application programs. Each may want to use a particular printer because of its location, speed, print quality, or all three.

Printer — A printer is a peripheral, which may be of three different types: system standard, system quality, or workstation printer. Several of each type may be distributed to different locations. Each printer is given a unique identifier at configuration time.

Unspooler — An unspooler is a program which controls the work of one type of printer. There are therefore three different unspooler types, one for each of the printer types named above. At configuration time, an "instance" (a copy) of the appropriate unspooler is provided for each printer. Each instance is provided with a unique path name.

Print job — This is a file (or set of files) for printing, known by a single job name. The user specifies the class to which it is to be added, according to the type of printer associated to the class. It is usually added to the class, then identified by a system-provided number, this being retained until the job has been printed. A print job also has a priority and a status. The priority indicates when it will be printed in relation to the other jobs in the queue, whilst the status indicates whether the job is currently waiting (HOLD), is ready for printing (READY), or is being printed (RUN).

OTHER UNSPOOLER INFORMATION

The workstation unspooler can only print for a user if that user has an active environment monitor on the terminal. It prints on the local printer of a workstation.

All unspoolers:

- print files as they find them (they are usually formatted before enqueuing, outside the spooling system)
- can perform the same print control functions
- print a header per job, if so configured, consisting of:
  - . date
  - . job identification number
  - . user name (owner of the job)
  - . job title, if supplied by the owner (omitted otherwise).

LINKS BETWEEN SPOOL OBJECTS

In order for the spooling system to print, links must be established between the printers, the jobs waiting to be printed, and the programs to print them. The following table shows the events and links created.

ACTION BY	STAGE	COMMAND	LINK CREATED
Installer/ System administrator	Configuration	SPCONF	Unspooler to printer
System administrator	Resource organisation	SPOOL	Class to unspooler
User	Operation	LPR	Job to class

Thus, during a work session, the user can add a job to a class, which has been linked to an unspooler already linked to a printer. The system also "knows" the printer location. Printing proceeds on a FIFO basis (First In, First Out).

The printer thus assigned is not available to print anything but jobs from that class, until deassigned (unless previous configuration allows the printing of jobs non submitted through SPOOL).

An unspooler can be called to operate concurrently on several separate printers. Hence the need for an unspooler instance for each printer.

CONTROL

Control of the spooling system is provided by the commands shown below, in the approximate order in which they are used. All their functions are available to the system administrator, and some to the normal user.

SPCONF configures the objects used to print through the spooling system.

SPOOL controls the operation of the spooling system.

LPR specifies the jobs to be printed and gives printing instructions.

LPQ displays the jobs in spooling classes awaiting printing.

System administration is a human activity whose purpose is to optimise the operation of a system. This responsibility requires further reaching command functions, as well as those available to "normal" users (see SPOOL command.)

Thus, the functions of the commands shown above fall into two categories:

1. Those intended for the system administrator to control the use of printing facilities in his network. This includes classes, printers, and individual jobs belonging to other users.

2. Those intended for the "normal" user to control, within the printing setup defined by the system administrator, only his own print jobs, and to access information on the work in progress.

PRINTING PROGRAM FILES THROUGH SPOOL

Print files from user programs may be directed to printers in one of the following ways:

- A program may access the printer directly by reference to its global name (for example, /DEV/SYSPRT1).

- Shell or BEAM may produce a print file for submission to the spooler.

- A program may access the printer and/or SPOOL directly, using the print commands of the language being used.

- Output from a program may be directed to a local name within the program which has been previously connected, with the command CONN or CONSP, to one of the predefined names SPOOL1 - SPOOL16. In this way, when the file is closed, the output file is queued in the appropriate SPOOL classes (see the commands CONN and CONSP in the manual MCL MOS Command Language User Guide).

DISTRIBUTED CONFIGURATION

The SPOOL system may be reconfigured by the system administrator to cater for changes in the printing capabilities of the system. Such changes are performed using SPCONF (see MOS System Software Generation and Installation Manual).

In a local network there can be spooling classes on different machines. Their existence and availability in the network is shown in the object table (see MOS Distributed System in Local Area Network (LAN) User Guide).

BATCH MANAGEMENT

The batch system manages job execution in background for a multi-user environment.

The user can normally execute jobs in suspend mode; this entails waiting for one process to finish before activating another.

The batch system allows users to enqueue and execute jobs without interrupting their current activity on the terminal.

The system administrator is able to control the operation of the batch system and to alter the status of any jobs in the batch class.

The batch system may be accessed directly by normal users to examine the state of the batch environment and to alter the status of their own jobs.

SUBMITTING JOBS

Jobs are submitted to the batch class by entering the command BM (see MCL MOS Command Language User Guide). The system informs the user of the unique identification number in the range 1 to 63 allocated to the enqueued job. Note that the BM command does not check whether the jobs submitted for queuing exist or not. The existence of a job is only checked when it is about to be executed.

USER ACCESS

The Shell command BATCH only allows the full range of functions to the system administrator. All other users have access to a subset of functions. Users may not change the status of a job submitted by another user.

Normal Users

All users of the system may:

- show the status of the batch class (STARTED or STOPPED)
- show the status of a job (HOLD, READY or RUN).

In addition the owner of a job may:

- set a job to HOLD (the job will not be executed until it is set to READY)
- set a job to READY (a previously held job will be executed when it has the highest priority in the class)
- kill a job (the job is removed from the class).

System Administrator

The system administrator may control the batch environment using the following restricted functions:

- start the batch class (jobs in the class will be executed)
- stop the class (jobs in the class will not be executed)
- kill a job submitted by a normal user, or change its priority or status.

SHELL VARIABLES

When a batch job is activated from a workstation, any Shell variables defined by the user in the command are supplied to the environment. The built-in variables, on the other hand, assume their standard value.

DISTRIBUTED CONFIGURATION

In a configuration distributed in a local network, there can be a batch environment on each machine connected to the network; however, there can only be one batch server configured and one job queue handled on the same machine. See MOS Distributed System in Local Area Network (LAN) User Guide.

PART 2 - COMMANDS

INTRODUCTION TO PART 2

This part (chapter 3) describes the Shell commands available to all users, in alphabetical order.

The method used to describe the command syntax is specified in chapter 3 of the DOCUMENTATION Guide.

3. DESCRIPTION OF THE SHELL COMMANDS

INTRODUCTION

This chapter contains details of all the commands available to normal users.

COMMAND SYNTAX

For commands which require parameters, if no parameters or incorrect parameters are given, a help string is displayed which indicates the syntax of the command:

USAGE: command param1 ... paramN (options) (**PRIVILEGED COMMAND**)

SYMBOLS USED IN THE EXAMPLES

The following graphic symbols are used in the examples in this chapter:

 indicates a file

 indicates a directory

 indicates a volume

* indicates the working directory

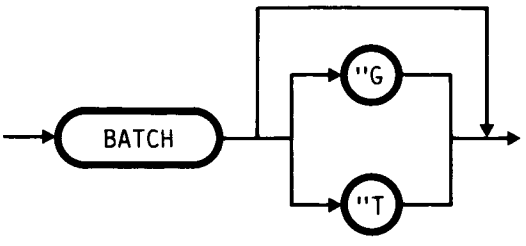
key indicates a key on the keyboard

BATCH is an interactive Shell command available to all users.

After entering the command, the user is presented with a menu of available functions (see the following figure). The functions available to normal users are:

- list the batch class (L)
- display the status of a job (J)
- set the status of a job to HOLD (H)
- set the status of a job to READY (R)
- remove (kill) a job from a class (K)
- return to the Shell environment (E).

All functions which affect individual jobs are only permitted to the owner of the job.



where:

"G is the option which allows the management of the jobs belonging to the global batch class in a distributed system.

"T is the option which allows the management of the jobs to be performed by a specified time.

The function types and the parameters are entered interactively (see "Operator Interface").

```
                    BATCH SYSTEM
                   --------------

      DO YOU WANT TO:

              L = LIST THE BATCH CLASS
              J = SHOW JOB STATUS
              H = SET HOLD A JOB
              R = SET READY A JOB
              K = KILL A JOB
              E = END

                  NEXT CHOICE >_
```

Fig. 3.BATCH-1 BACTH COMMAND - Normal User Menu

A batch function is selected by entering the letter corresponding to the required function and pressing **CR.** The menu is then replaced by a request for a parameter.

After a function has been performed the user may select another function by entering the corresponding letter.

At any time during parameter input or after a function has been performed the user may return to the main menu by entering:

. CR

Parameters

The parameters required by the BATCH functions are shown in the following table:

BATCH FUNCTION	COMMAND LETTER	PARAMETER REQUIRED
list the class	L	-
show job status	J	job number
set hold a job	H	job number
set ready a job	R	job number
kill a job	K	job number

Tab 1. BATCH Functions Parameters

The "job number" value is a number in the range 1..63, as allocated by BATCH when the job is submitted.

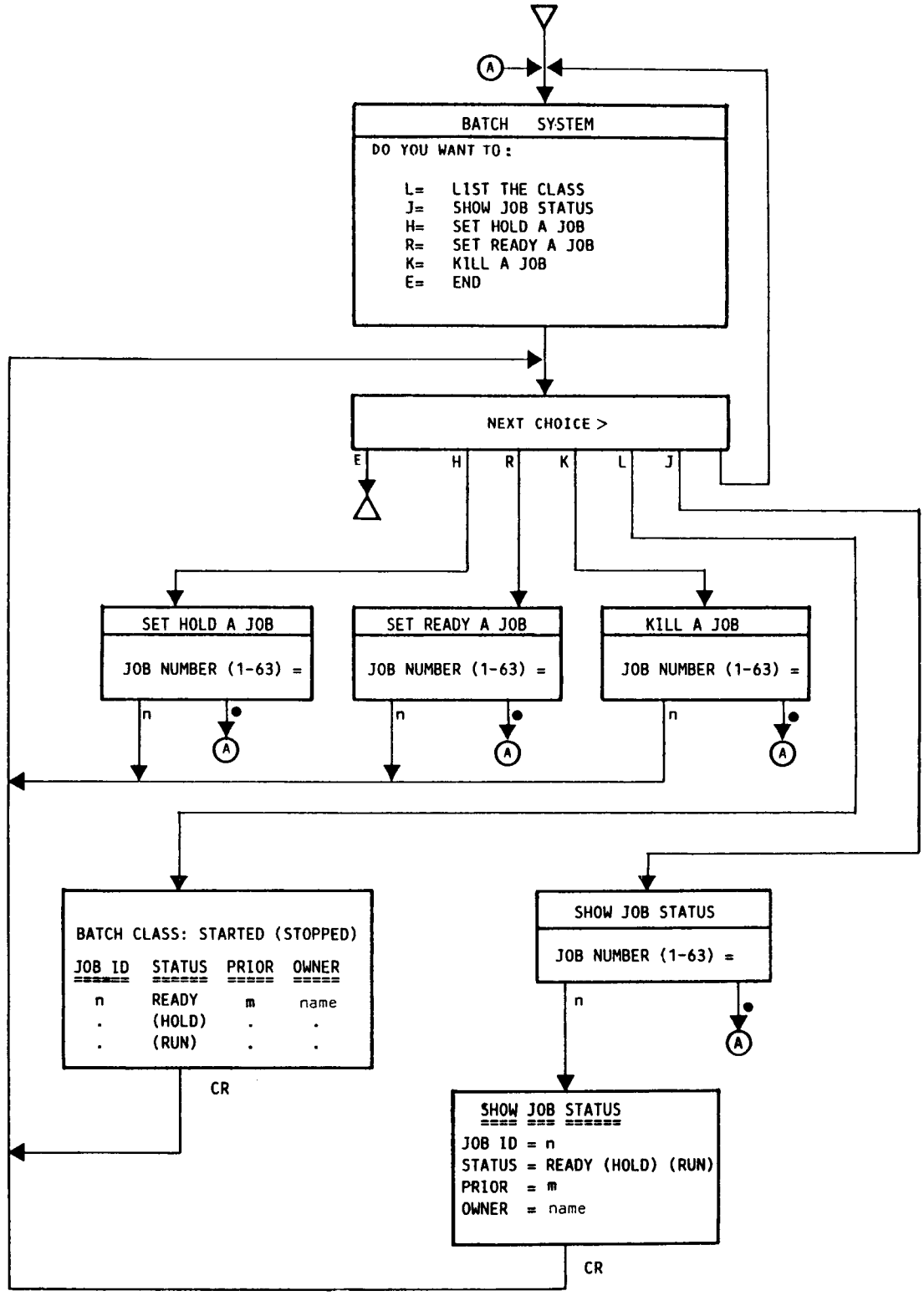


Fig. 3.BATCH-2 BATCH Command - Operator Interface

In the operator interface diagram shown:

JOBID is the job identification number allocated by the batch system.

STATUS is the current status of the job (either HOLD, READY or RUN).

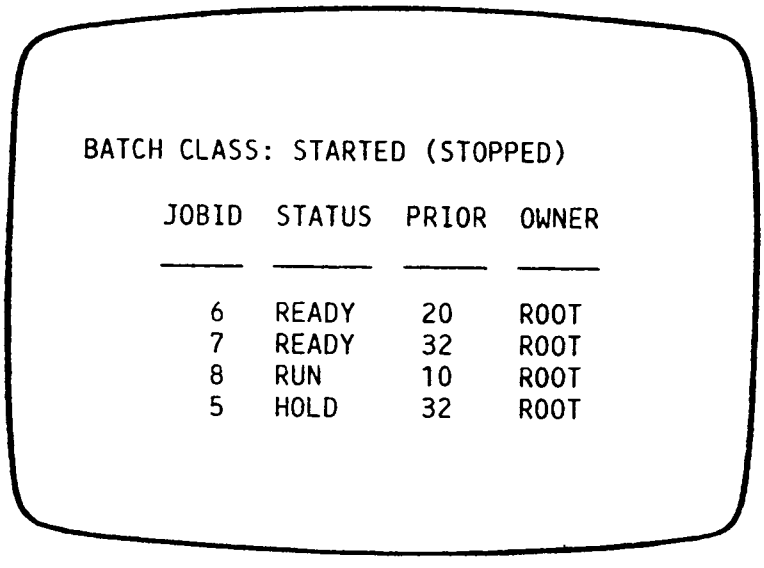
PRIOR is the priority of the job in the range 1 to 127. The job with the lowest priority number will be executed first. Jobs are initially given a priority of 32.

OWNER is the user name of the owner submitting the job.

When the batch class is listed (L) the jobs are listed in the following order:

- jobs with status READY, in order of priority or in order of submission
- the currently running job
- jobs with status HOLD, in the order in which their status was set to hold.

For example:



NEXT CHOICE >

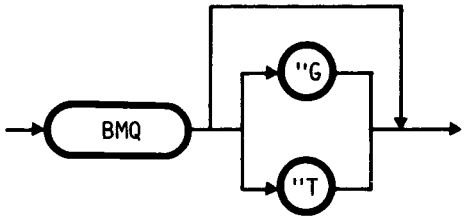
X [n]	"X" represents another BATCH function (the job number parameter if any can be specified directly).
.	Returns to the main menu.
E	Exits from the BATCH system and returns to Shell.

JOB NUMBER (1-63)=

n	Represents the required job number parameter.
.	returns to the main menu.

Displays the status of the batch class.

For an explanation of the information obtained, see the command BATCH in this chapter.



where:

"G is the option which allows the management of the jobs belonging to the global batch class in a distributed system.

"T is the option which allows the management of the jobs to be performed by a specified time.

Characteristics

In a distributed system, BMQ without any option displays the state of the local batch class.

Example

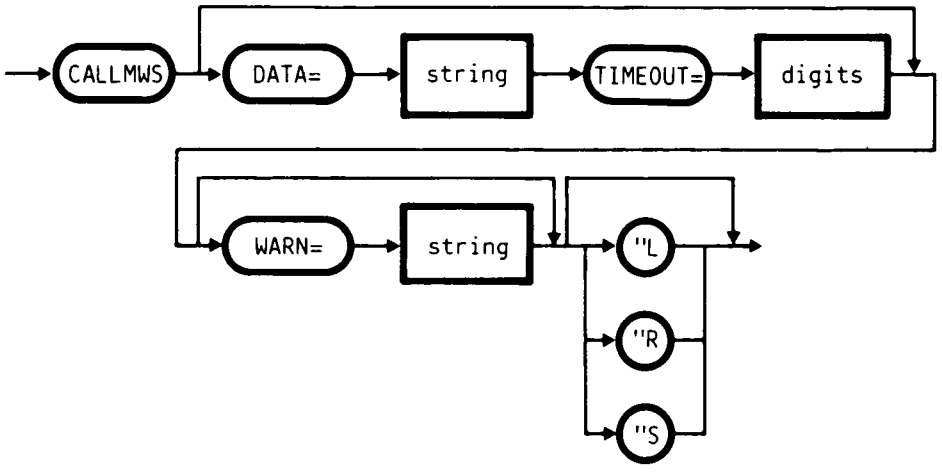
```
MCL: BMQ
BATCH CLASS: STARTED  (STOPPED)

        ID   STATUS    PRIOR   OWNER
        --   ------    -----   -----
        6    READY       20    ROOT
        7    READY       32    ROOT
        8    RUN         10    ROOT
        5    HOLD        32    ROOT

MCL:
```

Sends a request to the master workstation and may wait for a reply from the system manager (ROOT).

This command allows the user to send the following information to the local master workstation or, in a cluster configuration, to the local or global master workstation:

- a warning string
- a data string
- a data string, with a request for a reply.

If the user requests a reply, this will be loaded in the %RET variable if the option "S has been specified, otherwise it will be displayed on the screen.



where:

DATA string is the string of data to be displayed on the master terminal.

digits represent the maximum suspension time, expressed in milliseconds, that the caller will wait for a reply.

WARN string is the warning string that will be displayed on the warning terminal; this is made up of two parts: one defined by the user being the "WARN string" parameter, and another defined by the system, with the format "FROM user name (machine name)"

"L is an option identifying the local master workstation, as opposed to the global master workstation, which is assumed by default.

"R is the option with which to request a reply.

"S specifies that the reply must be stored as a string of characters in the %RET variable of the calling environment. (If this is omitted, the reply will be displayed on the standard output device.)

Characteristics

1. The data string that the user can send to the master workstation must have less than 250 characters. If the first character is "/" the string is interpreted as if it was the complete path name of a file, whose contents are to be displayed on the master workstation. In a distributed configuration the path name of the data must be global (e.g. /../machine name/etc).

2. Several options can be specified in the command line so that more than one service is performed (for example, **"LRS** requests an answer from the local master workstation with no message display in standard output).

3. If no workstation is specified as master during the configuration phase, one of the active workstations is selected to be the master workstation.

4. If the option **"R** is specified, the reply will be displayed in the following message:

   OPERATOR REPLY: xx...x

   where "xx...x" is the reply sent by the system administrator.

Note

See also Chapter 1.

Examples

1. The following command must be specified if a warning string is to be sent to a master workstation:

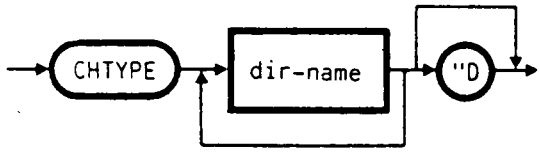
   ```
   CALLMWS WARN=string
   ```

2. If a data string is to be sent and an answer is required:

   ```
   CALLMWS DATA="restore tape and reply YES or NO" TIMEOUT=6000
           WARN="urgent message"  "R
   ```

Converts an ordinary directory into a program directory, and vice versa.

If no option is specified, a conversion from directory to program directory is assumed by default.



where:

dir-name is the name of the directory to convert.

"D specifies that the conversion is to be made from program directory to normal directory.

Characteristics

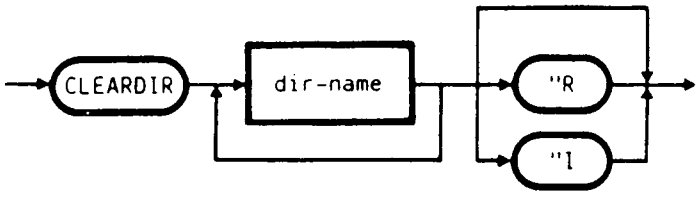
To execute this command, access rights for execution are required on both the normal and program directories. Only the primary owner and the system administrator are allowed to use the command.

Clears all the characters currently displayed on the screen, including the system line.

The visual attributes of the terminal are reset.

This command has no parameters.

Deletes the files in one or more specified directories. Only the directory files are cleared, not the sub-directories. If specified, the directory itself can also be deleted.



where:

dir-name is the name or path name of the directory.

"R is specified if the entire directory is to be cancelled.

"I specifies interactive command mode.

If interaction is specified the following message is displayed for each directory name or path name specified in the command line:

dir-name (Y/N)?

The user can now:

- press **Y** or **y** and the command is executed on the specified directory
- press **N** or **n** and the command will be ignored.

Characteristics

1. Neither "." nor ".." can be used as directory name.

2. If the option **"R** is specified in the command line and there are no active connections to nested directories, all the directories are deleted, including the root directory. It is not possible to delete in interactive mode.

3. All errors which can occur during the execution of CLEARDIR are displayed as follows:

   CLEARING path name: error message

   where "path name" is the name or path name of the file system object which has caused the error, and "error message" specifies the nature of the error.

4. As the CLEARDIR command involves remove operations, see the command REMOVE.

5. To execute this command, access right for writing is required on the directory or directories containing the files to delete.

Examples

We shall consider the memory structure shown here. If the following command is executed:

CLEARDIR /IPL/USR/FRANK/COB "R

the memory structure of a) will be modified to the memory structure of b), whereas the command:

CLEARDIR /IPL/USR/FRANK/COB

will modify the memory structure of a) to the structure of c).

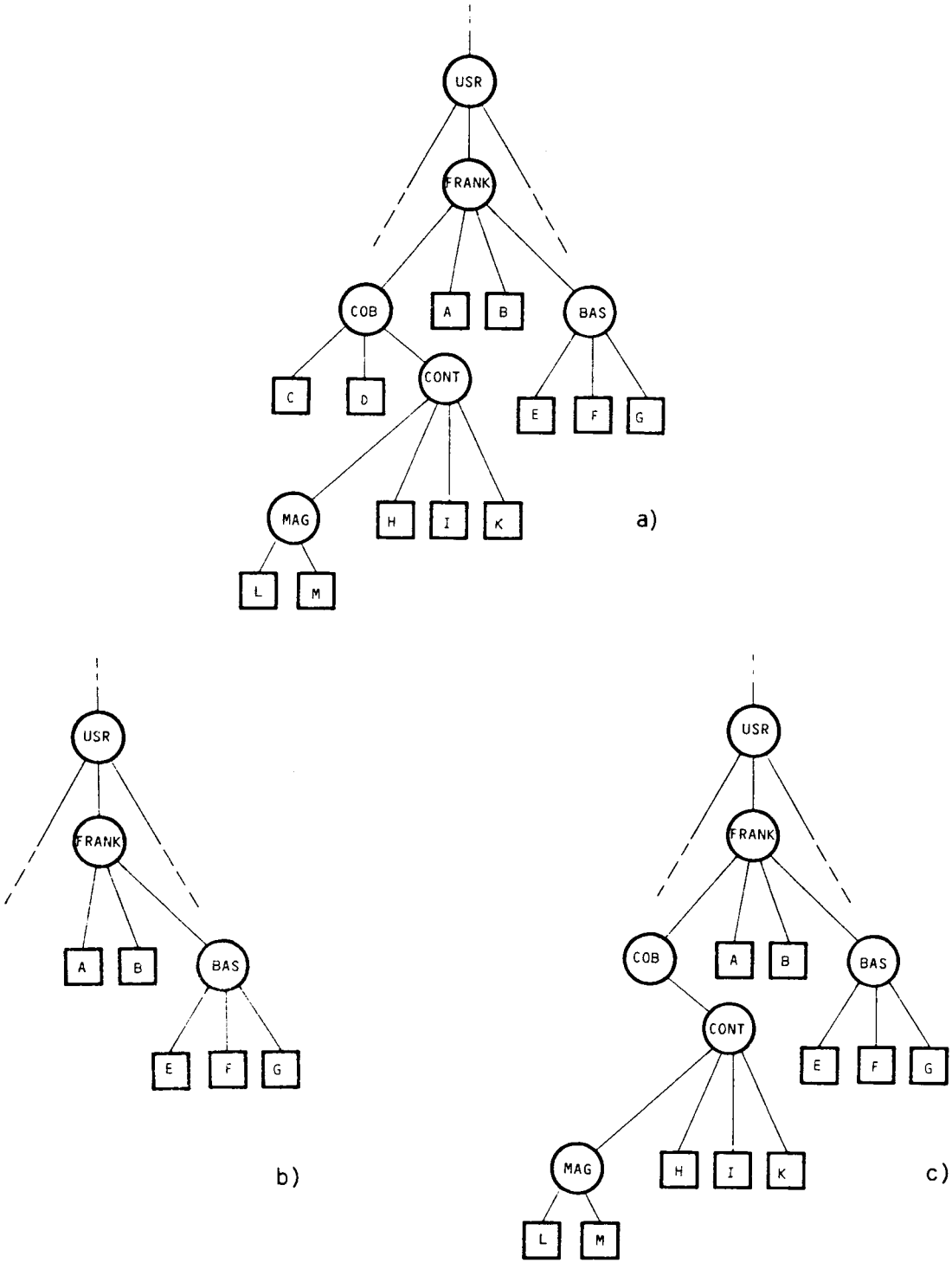
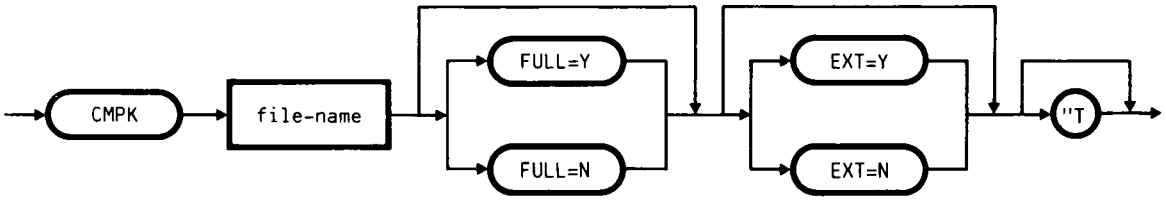


Fig. 3.CLEARDIR-1 CLEARDIR Example

CMPK compacts keyed files by removing the logical spaces that remain after some keys are cancelled. The maximum record length allowed is 30720 bytes. The maximum number of records is 1,198,080 (certified value of 650000 records).



where:

file-name is the name or path name of the file to compact, which must be local.

FULL=Y indicates that the index pages must be completely filled (fill factor 100%).

FULL=N indicates that the index pages must be filled to the amount given in the parameter SPLIT STRATEGY specified during creation of the file (see characteristics).

EXT=Y indicates that when compacting finishes the file extents must be kept the same as before.

EXT=N indicates that when compacting finishes the file extents must be truncated after the last significant record.

"T shows the total output; if this option is not specified, only the section headers are output (see below).

Characteristics

1. CMPK creates its temporary work areas in main memory only, without using any disc space, until the compacted file can be written back to disc.

   The amount of memory used depends on the size of the file to compact. There is an algorithm with which the user can calculate the amount of memory required (see "Limitations", further on).

2. If the key parameters FULL and EXT are not included in the command, **FULL=Y** and **EXT=N** are assumed by default.

3. The command requires access rights for reading and writing on the file to be compacted.

4. The compacted file is rewritten starting from the first original extent, i.e. in the same position on disc. A file can be compacted without freeing any disc space by specifying **EXT=Y**. If this is done, the file keeps its original extents, even if compacting results in unused space at the end of the file.

5. If **EXT=N** is specified, a file with multiple extents will occupy less extents after compacting, as the unused extents have been freed.

6. If **FILL=N** is specified, the fill factor is equal to the split strategy of the B*-tree pages defined when creating the file. The following table shows the relation between the split strategy and the fill factor:

SPLIT STRATEGY	FILL FACTOR
100	100%
90	90%
50	50%

7. The fill factor must be specified suitably, on the basis of how the file is to be used. If no write operations are to be carried out, it is preferable to specify a fill factor of 100%.

8. CMPK compacts the file in several different passes, which are executed by sections of the CMPK program.

SECTION 1 : COMPACT DATA FILE

Recovers the space occupied by the records that have been deleted logically, and ends the file after the last valid record.

SECTION 2 : UPDATE AND COMPACT LAST LEVEL OF KEY TREE

For each index, the key positions of each page of the last level are updated. The data on each page is compacted and then the pages are compacted according to the fill factor.

If **"T** is selected, the output from CMPK gives the value of the key, and the new and old positions of the record addressed by the key.

If **"T** has not been selected, only the section header is displayed.

SECTION 3 : REBUILD LAST LEVEL OF KEY TREE

The links between the pages of the last index level are rebuilt.

SECTION 4 : REBUILD KEY TREE

The upper levels of the B*-tree are rebuilt for each index, and the space which is not used by the file index is freed.

Limitations

Before using CMPK on a file, the user should check that it is not too large for the memory space available, bearing in mind the following:

1. Three segments of memory are available, the size of each being 1024 x 64 bytes.

2. The memory required for the compaction operation is referred to in the algorithm as "R2".

3. The memory R2 is freed at the end of section 1.

R2 can be calculated as follows; given:

Number of file records to compact = R

and:

R1 = (R / 256) + 1 if ((R mod 256) <> 0)

(rounding the result of dividing R by 256 up to the nearest integer)

the amount of memory required is:

R2 = R1 * 38 (bytes)

Notes

1. Before activating CMPK a check must be made that the file structure is consistent, using the command KEYCHECK (see MOS System Software Maintenance Tools User Guide).

2. Before a file is compacted, a backup copy must be made.

3. Execution of CMPK must not be interrupted, otherwise the file data will not correspond to the actual situation.

4. The output comments from CMPK appears on the screen unless they are redirected to a printer, which is advisable.

Example

An example of a complete CMPK output follows:

If shortened output is requested, i.e. the option "T is not used, only the headings of each section are displayed.

```
   SECTION 1: COMPACT DATA FILE

     *** TRACE OF FIELD NUMBER : 0 ***

   SECTION 2: UPDATE AND COMPACT LAST LEVEL OF KEY TREE

     *** UPDATE PAGE AT OFFSET : 0 ***
    KEY          OLD POSITION     NEW POSITION

0178109335                  18
0390009625                  20
0400912534                  17
0454585499                  22
0515080363                  18
0613406366                  15
0659470878                  13
0705557792                   1
0711281189                   7
0803056554                   3
0822641953                   9
0845081645                   4
0894113186                   8
0919117722                   2
0958123088                   6
1051878933                  16
1170815012                  14
1188173609                  11
1191207346                   5
1194408627                  12
1280727471                  19
1319880733                   9
1501267736                   4
1523967651                  15
1560960166                  10
1607373876                   7
1641349806                  21
1647882151                  17
1820009887                  16
2005086510                  18
2020857388                   3
2089049108                  15
2123399976                  22

   SECTION 3: REBUILD LAST LEVEL OF KEY TREE

   SECTION 4: REBUILD KEY TREE .
```

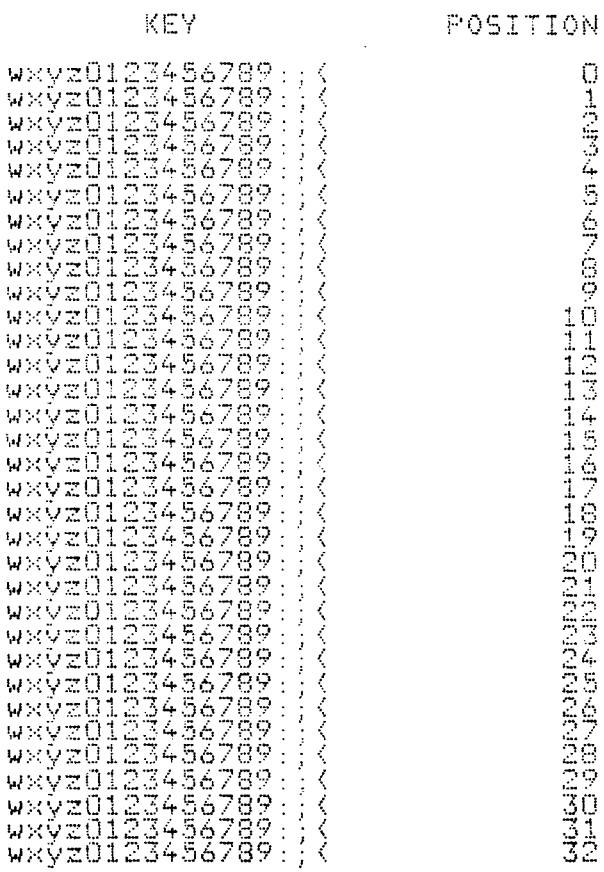
Fig. 3.CMPK-1 CMPK output (cont.)

```
*** TRACE OF FIELD NUMBER : 1 ***

SECTION 2: UPDATE AND COMPACT LAST LEVEL OF KEY TREE .

   *** UPDATE PAGE AT OFFSET : 0 ***
   KEY          OLD POSITION     NEW POSITION

CD0178109335                 18
CD0390009625                 20
CD0400912534                 17
CD0454585499                 19
CD0515080363                 18
CD0613406366                 15
CD0659470876                 13
CD0020555779                 14
CD0711281180                 16
CD0803056554                 17
CD0822641953                 12
CD0845081645                 10


CD0894113186                 29
CD0919117722                 11
CD0958122289                 16
CD1051879933                 16
CD1170815013                 11
CD1188173609                 16
CD1191207346                 15
CD1194408627                 16
CD1280727471                 16
CD1319880733                 19
CD1501267736                 19
CD1523967651                 30
CD1560960166                 11
CD1607273876                 17
CD1641349806                 14
CD1647882151                 14
CD1822000098                 17
CD2005086513                 16
CD2022085738                 16
CD2069049108                 15
CD2123399976                 15
   SECTION 3: REBUILD LAST LEVEL OF KEY TREE .

   SECTION 4: REBUILD KEY TREE .
```

Fig. 3.CMPK-2 CMPK output (cont.)

```
   *** TRACE OF FIELD NUMBER : 2 ***

  SECTION 2: UPDATE AND COMPACT LAST LEVEL OF KEY TREE .

   *** UPDATE PAGE AT OFFSET : 0 ***
        KEY              OLD POSITION     NEW POSITION

wxyz0123456789:;<             15                0
wxyz0123456789:;<             16                1
wxyz0123456789:;<             17                2
wxyz0123456789:;<             18                3
wxyz0123456789:;<             19                4
wxyz0123456789:;<             20                5
wxyz0123456789:;<             21                6
wxyz0123456789:;<             22                7
wxyz0123456789:;<             23                8
wxyz0123456789:;<             24                9
wxyz0123456789:;<             25               10
wxyz0123456789:;<             26               11
wxyz0123456789:;<             27               12
wxyz0123456789:;<             28               13
wxyz0123456789:;<             29               14
wxyz0123456789:;<             30               15
wxyz0123456789:;<             31               16
wxyz0123456789:;<             32               17
wxyz0123456789:;<             33               18
wxyz0123456789:;<             34               19
wxyz0123456789:;<             35               20
wxyz0123456789:;<             36               21
wxyz0123456789:;<             37               22
wxyz0123456789:;<             38               23
wxyz0123456789:;<             39               24
wxyz0123456789:;<             40               25
wxyz0123456789:;<             41               26
wxyz0123456789:;<             42               27
wxyz0123456789:;<             43               28
wxyz0123456789:;<             44               29
wxyz0123456789:;<             45               30
wxyz0123456789:;<             46               31
wxyz0123456789:;<             47               32
   *** UPDATE PAGE AT OFFSET : 1024 ***
        KEY              OLD POSITION     NEW POSITION

SYNCRONIZE BUFFER PAGE AT FILE POSITION : 0 FOR LENGTH : 1024

HEADER :

BYTES USED   : 707
NEXT PAGE    : 1024
PREV. PAGE   : -1
```

Fig. 3.CMPK-3 CMPK output (cont.)



SECTION 3: REBUILD LAST LEVEL OF KEY TREE .

SECTION 4: REBUILD KEY TREE .

Fig. 3.CMPK-4 CMPK output

Information Messages

*** TRACE OF FIELD NUMBER : n ***

"n" indicates the index page number.

KEY

Each key is listed in its page order.

OLD POSITION

The position on the page of a record associated to a particular key before the file is compacted.

NEW POSITION

The position on the page of a record associated to a particular key after the file is compacted.

SYNCHRONIZE BUFFER PAGE AT FILE POSITION : 0 FOR LENGTH : 1024

Header showing that the current page is the first in the index, and is 1024 bytes long.

HEADER

Page header, made up of the three following values.

BYTES USED

Number of bytes filled by data on the page.

NEXT PAGE

Initial address of the page following the current one, expressed as a byte offset from the beginning of the index. If there is no further page, the value is -1.

PREV. PAGE

Initial address of the page preceding the current one, expressed as a byte offset from the beginning of the index. If there is no previous page, this value is -1.

KEY and POSITION

List of all the keys and their positions.

DATA FILE ALREADY COMPACTED

None of the keys in the file have been deleted, so compacting cannot take place.

ERROR IN READ INFO FILE

Error reading the info file.

ERROR IN READ/WRITE DATA FILE

An error has occurred when reading or writing the data file for one of the following reasons:

- A hardware error has occured.
- There is no space on disc or in the system tables.
- The file name has been entered wrongly.
- The file protection does not allow access of this type.
- The contents of the index are not consistent with the data file.
- The structure of the index is incorrect.
- A record cannot be accessed because it is being used by another process.

ERROR IN READ/WRITE/REWRITE/UPDATE INDEX FILE

The file index cannot be read or updated for one of the reasons described in the previous message.

ERROR IN REWRITE KEY OFFSET IN INDEX FILE

The index cannot be updated for one of the following reasons:

- The key structure is inconsistent.
- Some of the keys do not exist.
- Some of the keys have been duplicated.

ERROR IN REWRITE INFO FILE

An error occurred while rewriting the info file for one of the reasons described above.

ERROR IN REWRITE LAST INDEX PAGE

An error occurred while rewriting the last index page for one of the reasons described above.

ERROR IN TRUNCATE DATA/INDEX FILE

The data/index cannot be truncated for one of the following reasons:

- A hardware failure has occurred.
- A system error has occurred or a system table is full.

REMOTE FILE : NO REMOTE ACTION
INVALID OPERATION

CMPK cannot operate on remote files.

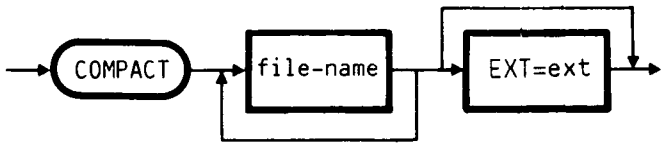
WARNING: ALL KEYS DELETED TRUNCATE DATA FILE AND REBUILD INDICES

All the keys in the file to be compacted have been deleted: in this case, CMPK truncates the data file and rebuilds the file indices without going through the steps just described.

WRONG INDEX : INVALID RECORD POSITION

A position has been found out of the file boundaries, for an index file incostistency. The user would have to issue a KEYCHECK before CMPK. The index is no longer recoverable.

COMPACT compresses the component parts of each file specified, into a single storage space if possible. This reduces the file access time, most of all because it reduces movement of the read/write head of the disc unit.



where:

file-name specifies the path name of the file to compact.

ext specifies the size required for the destination extent, in bytes; this assumes by default the total current file size. It may be more than is currently needed, taking account of anticipated requirements. However, the space in excess would then be wasted.

Caution: ensure that a backup copy exists before attempting to compact a file. A system crash during compaction would damage the original file.

Characteristics

1. The COMPACT command can only be used in non-interactive mode.

2. The storage space from previously logically deleted records will be freed.

3. Temporary disc files (support files) are created during the compacting process.

4. A compacted file is written starting at the position of the first extent of the original. If there is enough space, the entire file is written contiguously, otherwise the other extents are placed wherever space is found.

5. Byte-stream or positional files (non-keyed files): first, COMPACT looks for a contiguous area of memory big enough to hold the file, and if found, uses it to compact the fragments of the file. It is only if the file is too big to fit in memory that disc work areas are used.

6. Keyed files are always compacted using support disc files. Valid records and keys are transferred to a support file, the primary index to another, and any secondary index to yet another. When data transfer is complete, each support file is compacted, and then given the name of the original file.

7. COMPACT cannot handle positional files with record deletion not permitted: if one of these requires compacting, use COPY.

8. The allocation unit of an index file is not changed during compaction.

9. The following access rights are required for executing the command:

   - append or execution and write on the father directory of the file to compact

   - read and write on the file to compact.

10. All the errors that occur during execution of COMPACT are displayed as follows:

    COMPACTING path name : error message

    where "path name" is the name or path name of the file system object that has caused the error, and "error message" describes the nature of the error.

Optimisation

If used sensibly to give a file only as much space as it needs, COMPACT tends to leave a significant amount of the released disc space free. Users should periodically:

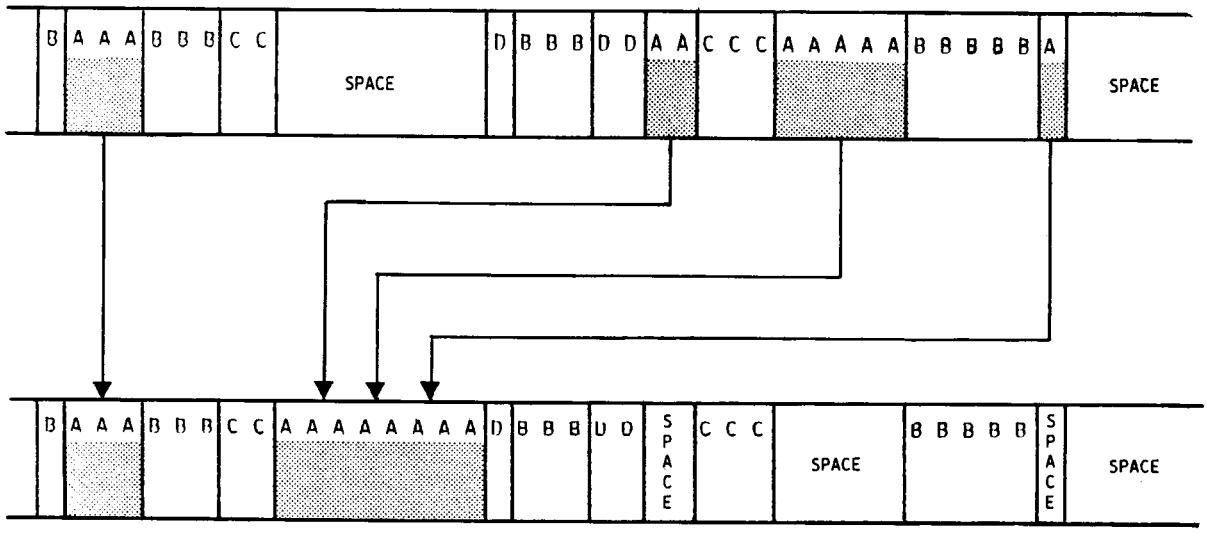
1. Create a support volume (on hard or floppy disc), with the same parameters as the volume to compact.

2. Copy all the first level volumes, directories and files to the support volume, with the command **CPTREE "R.** This ensures that the entire volume is copied in an unfragmented manner.

3. Copy the support volume back to the original volume. After this activity the volume is organised as shown in the following figure.

4. A (possibly risky) alternative to issuing many single COMPACT commands is to run VCOMPACT once.

Before COMPACT, file A's extents appear thus:



After COMPACT has been performed on file A, its extents appear as above.

COPY may be used for further reorganisation. For example, the following could be achieved by 4 separate COPY runs:

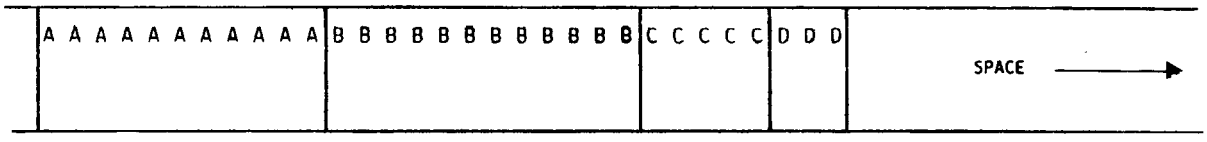


Fig. 3.COMPACT-1 Effect of COMPACT

Converts a file from one type to another by specifying the original type, the new type and any necessary parameters.

The file types that can be converted are:

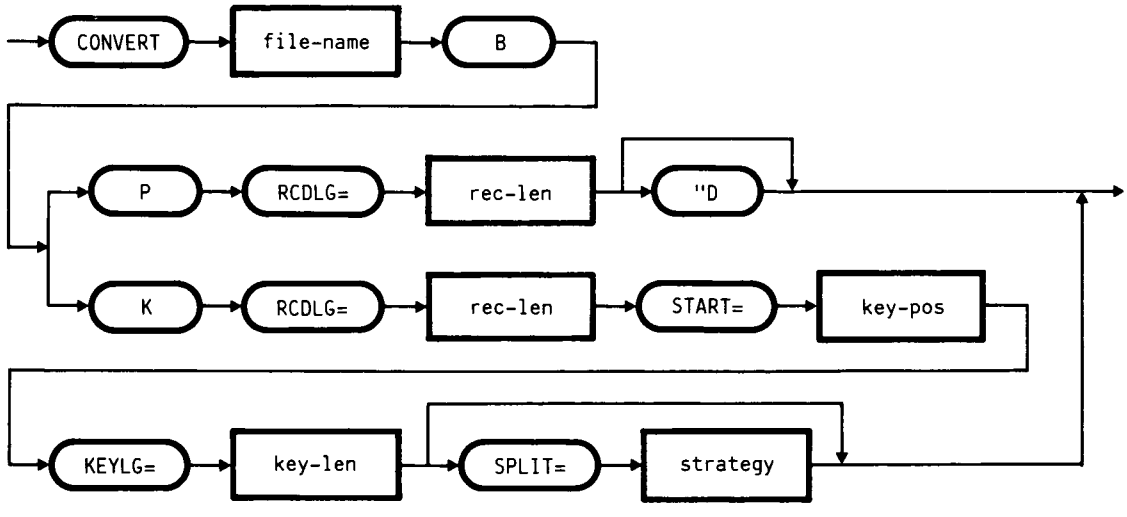
- byte-stream (B)
- positional with record deletion permitted (P)
- positional with record deletion not permitted (P)
- keyed (K).

The possible conversions for each type of file are described here.

BYTE-STREAM FILES

A byte-stream file can be converted to:

- a positional file with record deletion not permitted, by specifying only the record length
- a positional file with record deletion permitted, specifying the appropriate option as well as the record length (see MKPOS)
- a keyed file, specifying the record length, the offset of the first byte of the key with respect to the start of the record, the length of the key itself and, optionally, the split strategy of the B*-tree page (see MKKEYED command).



where:

file-name is the name or path name of the file to be converted.

B indicates that the file to be converted is a byte-stream file.

P or **K** indicates that the file is to be converted to a positional or keyed type.

"D specifies that the file is to be converted into a positional file with record deletion permitted.

rec-len specifies the record length expressed in number of bytes. The size of the byte-stream file must be a multiple of the record length.

key-pos specifies the offset of the first byte of the key with respect to the start of the record, which has offset 0. The key position value must be greater than or equal to zero.

key-len specifies the key length expressed in number of bytes.

strategy is the split strategy of the B*-tree pages: the accepted values are 50, 90, 100; if the parameter is not specified, the default value of 50 is assumed.

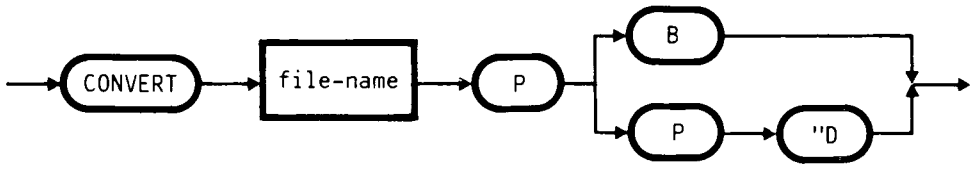
Note

There is a physical limit to the size of a byte-stream file which may be converted into a keyed file on a 1 Mbyte floppy disc. This is because a keyed file occupies more disc space. In practice, the limit is 890 Kbytes.

POSITIONAL FILES WITH RECORD DELETION NOT PERMITTED

A positional file with record deletion not permitted can be converted into:

- a byte-stream file, without any parameters
- a positional file with record deletion permitted, by specifying the appropriate option.



---

where:

file-name is the name or path name of the file to be converted.

P specifies that the file to be converted is of positional type with record deletion not permitted.

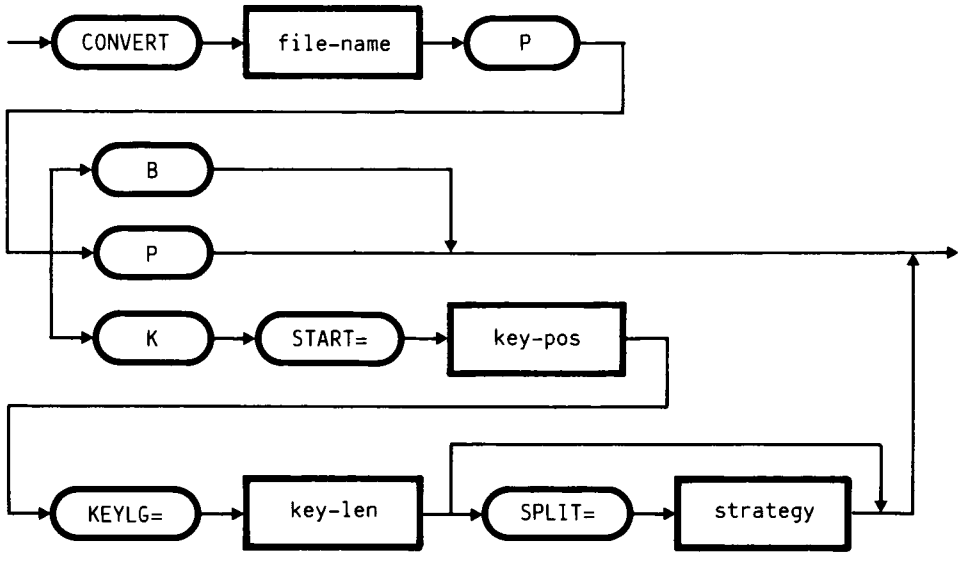
B specifies that the file is to be converted into a byte-stream file.

P, "D specifies that the file is to be converted into a positional file with record deletion permitted.

POSITIONAL FILE WITH RECORD DELETION PERMITTED

A positional file with record deletion permitted may be converted into:

- a byte-stream file, without any parameters
- a positional file with record deletion not permitted, without any parameters
- a keyed file, specifying the offset of the first byte of the key with respect to the start of the record, the length of the key itself, and optionally, the split strategy of the B*-tree page.



where:

file-name is the name or path name of the file to be converted.

P specifies that the file to be converted is of positional type with record deletion permitted.

B, P or **K** specifies that the file is to be converted into a byte-stream, positional with record deletion not permitted, or keyed file respectively.

key-pos specifies the offset of the first byte of the key with respect to the start of the record, which has offset 0. The key position value must be greater than or equal to zero.

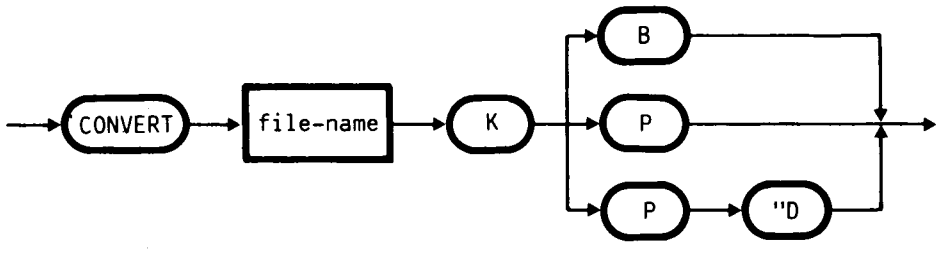
key-len is the key length expressed in bytes.

strategy is the split strategy of the B*-tree pages: the accepted values are 50, 90, 100; if the parameter is not specified, the default value of 50 is assumed.

KEYED FILE

A keyed file may be converted into:

- a byte-stream file, without any parameters
- a positional file with record deletion not permitted, without any parameters
- a positional file with record deletion permitted, by specifying the appropriate option.



---

where:

file-name is the name or path name of the file to be converted.

K specifies that the file to be converted is keyed.

B specifies that the file is to be converted into a byte-stream file.

P specifies that the file is to be converted into a positional file with record deletion not permitted.

P, "D specifies that the file is to be converted into a positional file with record deletion permitted.

Characteristics

1. The command will only accept the following file types: B (byte-stream), P (positional), K (keyed). If any other character or string is entered an error message is displayed.

2. Only the conversions described are allowed. Any other conversion attempt will cause the display of an error message.

3. As this command may involve removal of auxiliary files, see the REMOVE command.

4. It is possible to convert a keyed file with an external primary key into a positional file. However, it is not possible to recover the original keyed file because the key once lost cannot be recreated.

5. The following access rights are required for perfoming the command:

   - execution and read, or append on the father directory of the file to convert

   - read and write on the file to convert.

Examples

1. CONVERT /IPL/USR/BOB/COBOL/DATA1 B K RCDLG=512 START=78 KEYLG=20

   The byte-stream file DATA1 is converted into a keyed file of the same name. The record length of the new file is 512 bytes, the key is placed starting at byte 78 and is 20 bytes long.

2. CONVERT BASIC/PROG K P "D

   Converts the keyed file PROG into a positional file with record deletion permitted of the same name. The record length of the positional file is the same as that of the keyed file.

This command, issued in a MOS environment at release 5.2, converts a packed positional file to a positional file with record deletion permitted.

where:

input-filename is the file name (in the working directory) of the packed positional file to be converted; it cannot be the name of an alias file.

output-filename is the file name (in the working directory) of the output positional file with record deletion permitted; it cannot be the name of an alias file. If a file with that name already exists, an error occurs.

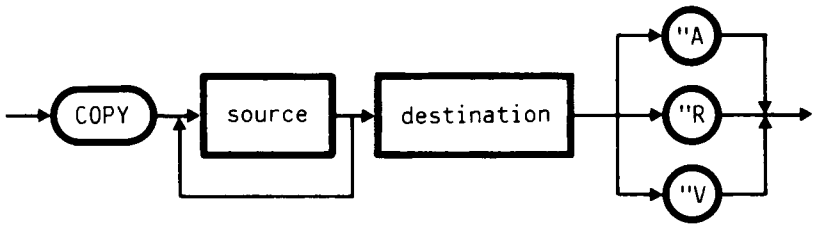
Copies a source file, directory, or volume into a corresponding destination file, directory, or volume.

The source and destination files, directories, or volumes can reside on different drives.

The command gives different results depending on the source and the destination, allowing the user to:

- copy a source file onto a destination file, directory or volume
- add a source byte-stream file onto the end of a destination byte-stream file
- copy a source file list onto a destination directory or volume
- copy a source directory onto a destination file or directory
- copy a source volume onto a destination file or volume.

Each of these cases is described below. The general format of the command is:



where:

source is the name or path name of the source file, directory, or volume.

destination is the name or path name of the destination file, directory, or volume.

"A is the option for appending a byte-stream file to another byte-stream file.

"R is the replace option.

"V is the verbose option: the name of each item copied is displayed.

Characteristics

1. Errors which occur during the execution of the COPY command are displayed in the following format:

   COPYING path name: error message

   where "path name" is the name or path name of the source file system object which has caused the error and "error message" describes the nature of the error (see Appendix A).

2. The file to which the source file is to be appended must not have an allocation unit (AU) of zero.

3. As the COPY command with the option "R involves remove operations, see the command REMOVE.

4. The following access rights are required to execute this command:

   - execution on the father directory of the source object
   - read on the source object
   - write or append on the destination directory.

5. Both when the destination object is created, and when it exists already, it is assigned the access rights defined in the default access list of the user performing the operation.

COPYING A FILE

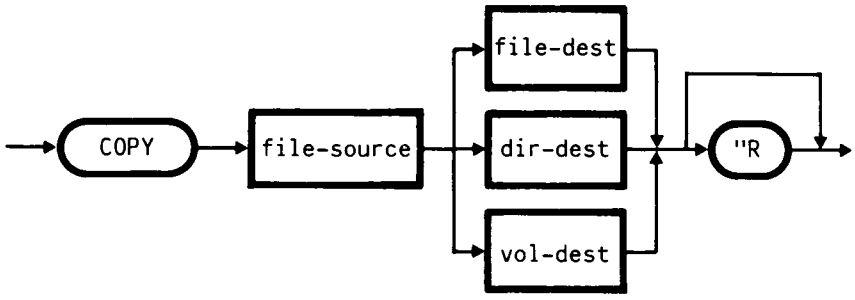
A source file may be copied onto:

- a destination file

  If the destination file does not exist, it is created and the source file is copied over it. If it already exists, the replace option copies the source file overwriting the destination file.

- an existing destination directory or volume.

  A copy of the source file is created and this is inserted into the first level of the destination directory or volume. If a file system object of the same name as the source file already exists in this level, it will be overwritten by the source file if the replace option has been specified, otherwise the command is aborted.



where:

file-source is the name or path name of the source file.

file-dest is the name or path name of the destination file.

dir-dest is the name or path name of the destination directory.

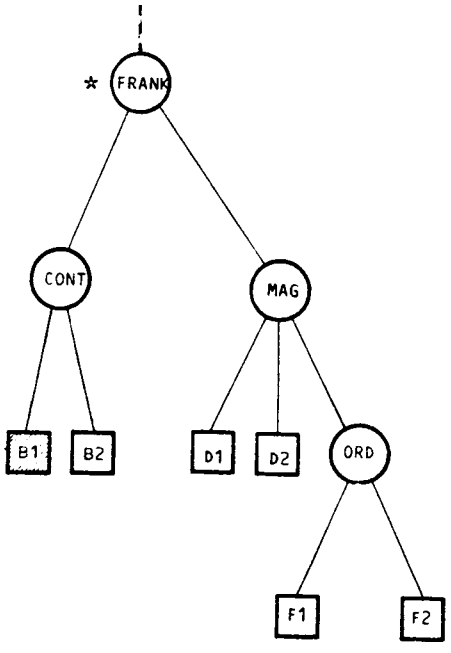
vol-dest is the name or path name of the destination volume.

"R is the replace option.

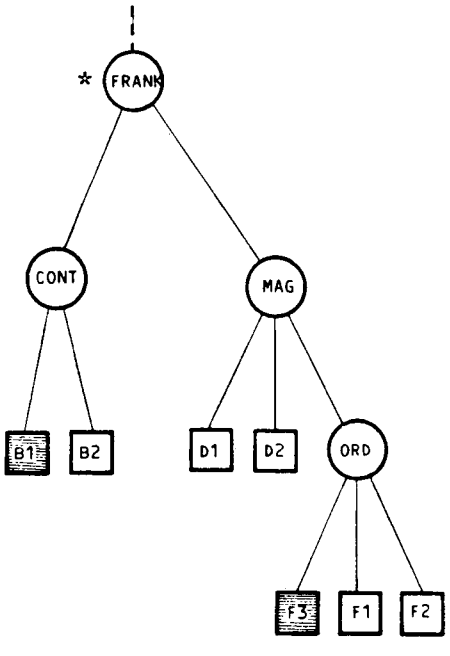
Examples

1. COPY CONT/B1 MAG/ORD/F3

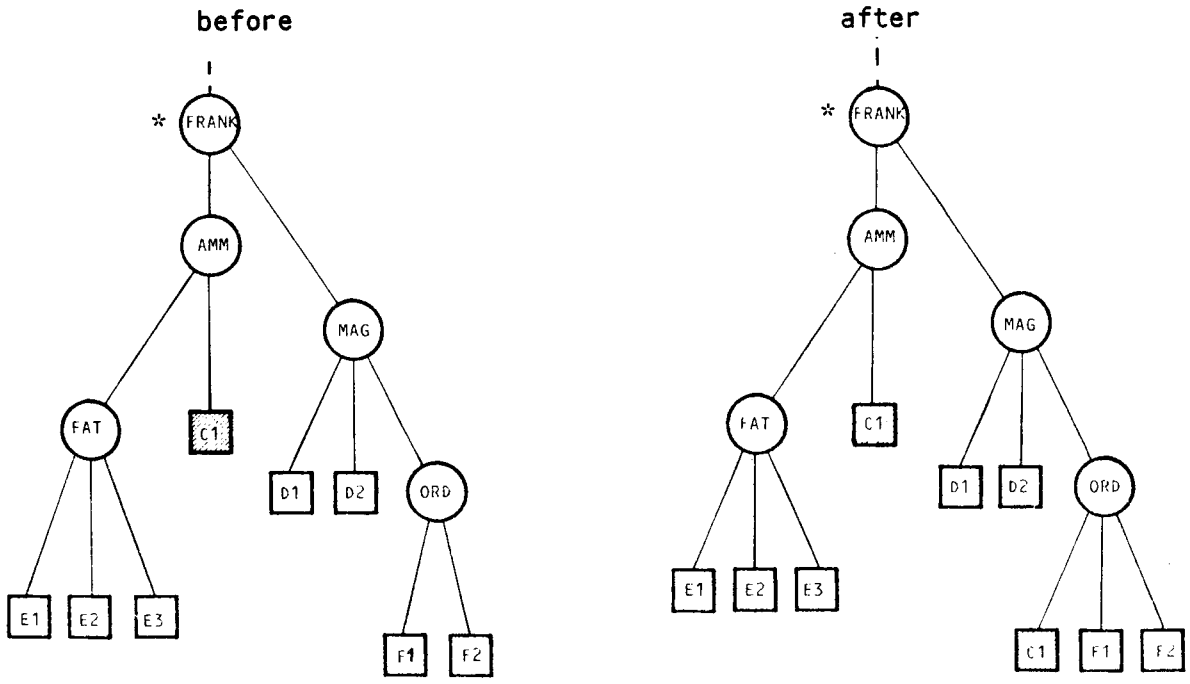
before



after

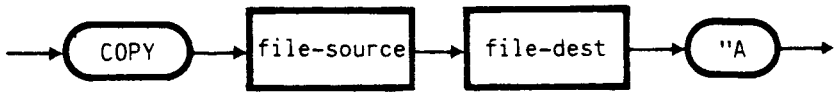


2. COPY AMM/C1 MAG/ORD



APPENDING A BYTE-STREAM FILE

A source byte-stream file may be appended to a destination byte-stream file when the append option is specified.



where:

file-source is the name or path name of the source byte-stream file.

file-dest is the name or path name of the destination byte-stream file.

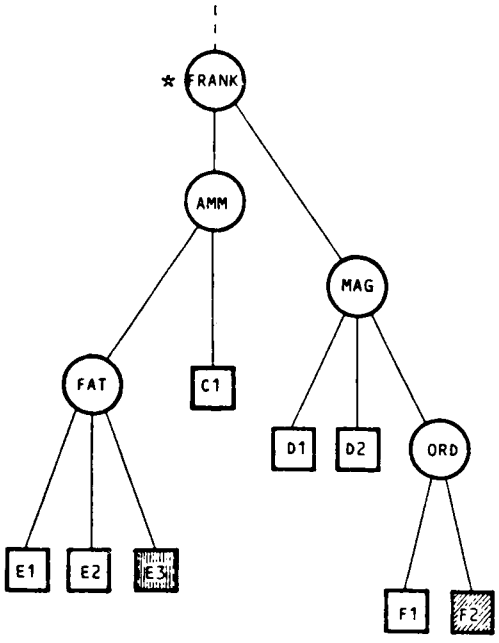
"A is the append option.

Examples

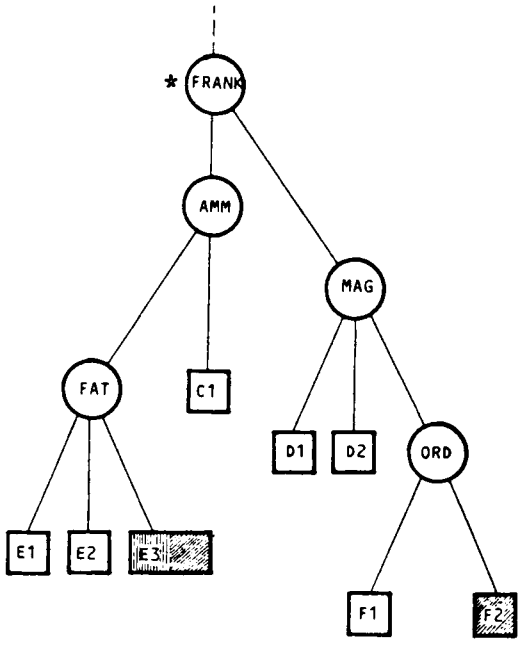
1. The files F2 and E3 are byte-stream files.

```
COPY MAG/ORD/F2 AMM/FAT/E3 "A
```

before



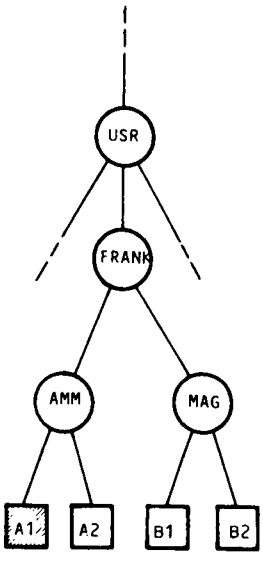
after



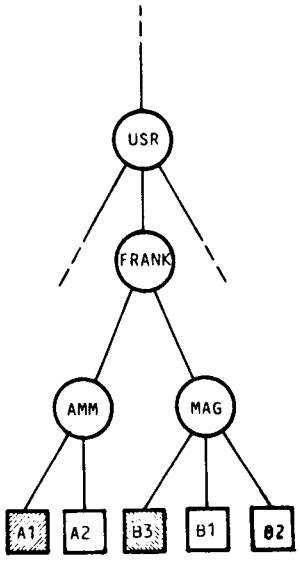
---

2. COPY FRANK/AMM/A1 FRANK/MAG/B3 "A

before



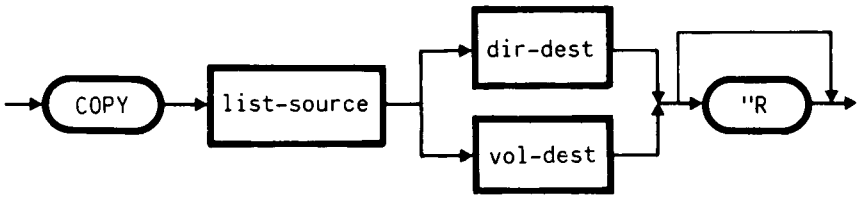
after



COPYING A FILE LIST

A source file list may be copied onto an already existing destination directory or volume. A copy of each file in the source file list is created, and inserted into the first level of the destination directory or volume.

If any of the source file or volume names already exist on the destination volume or directory, they will be overwritten by the corresponding source file only if the replace option was specified, otherwise an error is signalled.



where:

list-source is the list of the names and/or path names of the source files.

dir-dest is the name or path name of the destination directory.

vol-dest is the name or path name of the destination volume.

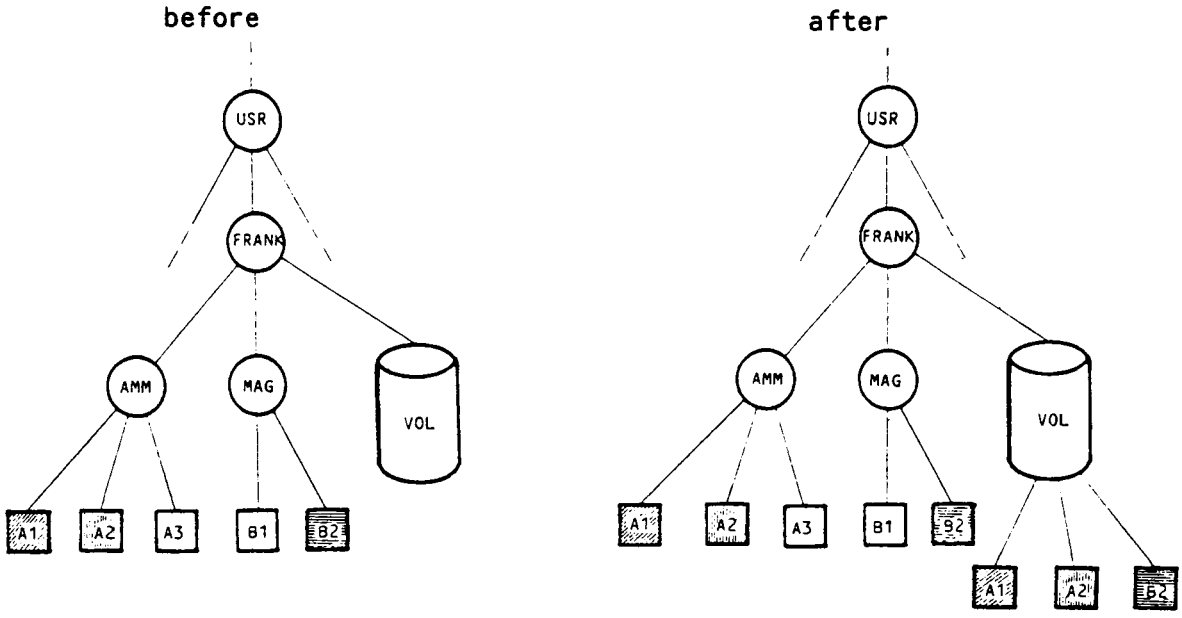
"R is the replace option.

Characteristics

If the user specifies a file name as destination, an error message is displayed (see Appendix A).

Example

COPY FRANK/AMM/A1 FRANK/AMM/A2 FRANK/MAG/B2 FRANK/VOL



COPYING A DIRECTORY

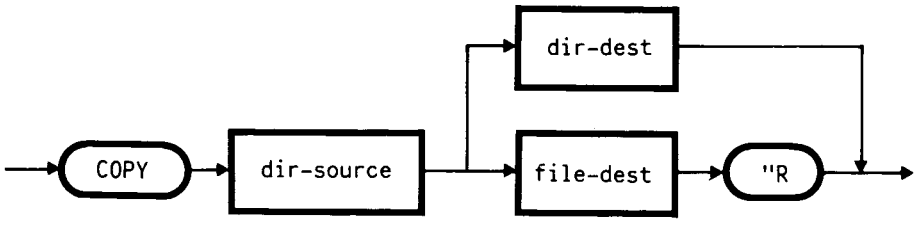
A source directory may be copied onto:

- a non-existent name

  A directory with the name specified is created and all the files, volumes, and sub-directories of the source directory are copied in this directory.

- a destination file

  When specifying the replace option, the destination file is overwritten by the source directory, which takes the name of the destination file.



---

where:

dir-source is the name or path name of the source directory. It cannot be "." (working directory) or ".." (father directory).

file-dest is the name or path name of the destination file.

dir-dest is the name or path name of the destination directory.

"R is the replace option.

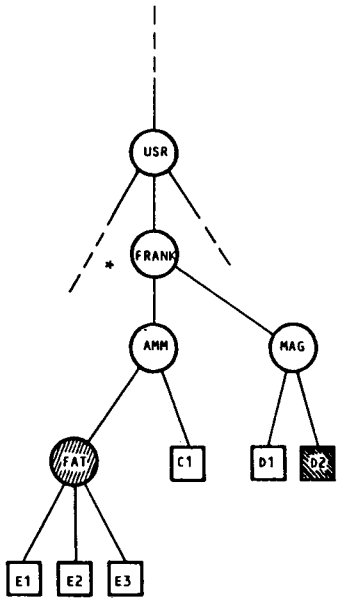
Characteristic

The above command operates also on a program directory.

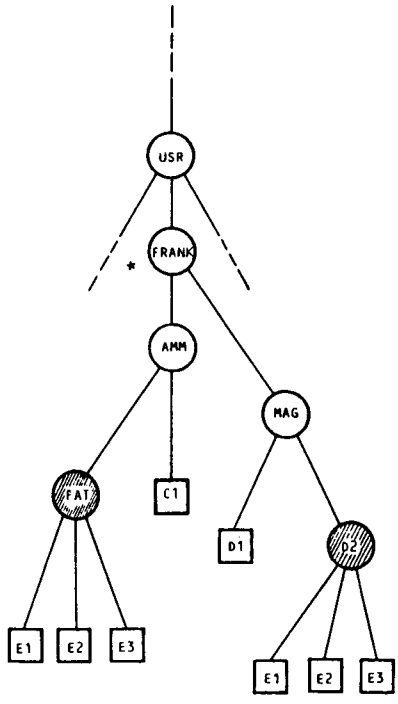
Examples

1. COPY AMM/FAT MAG/D2 "R

before



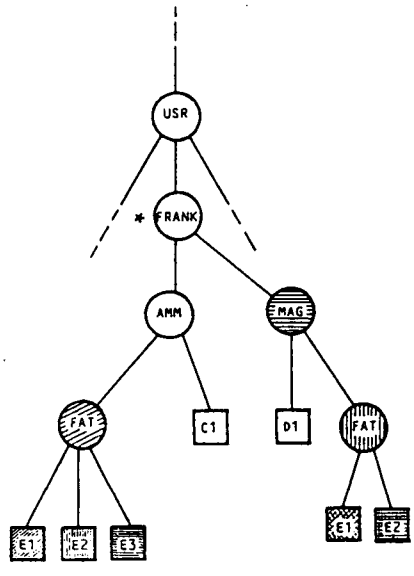
after



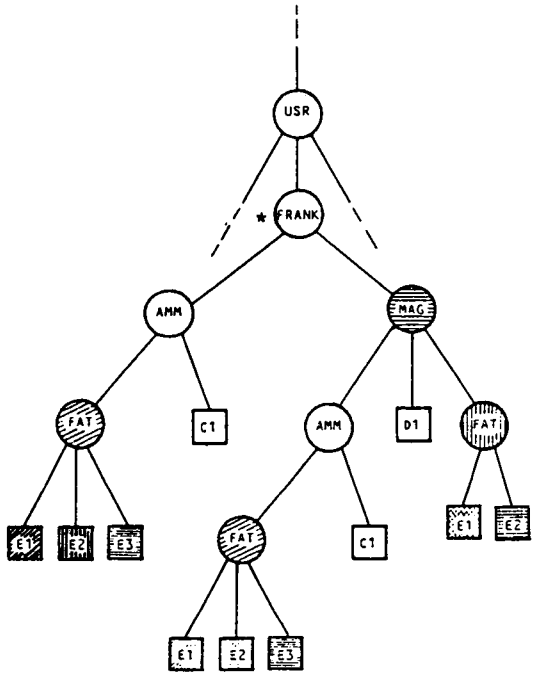
---

2. COPY AMM MAG/AMM

before



after



COPYING A VOLUME

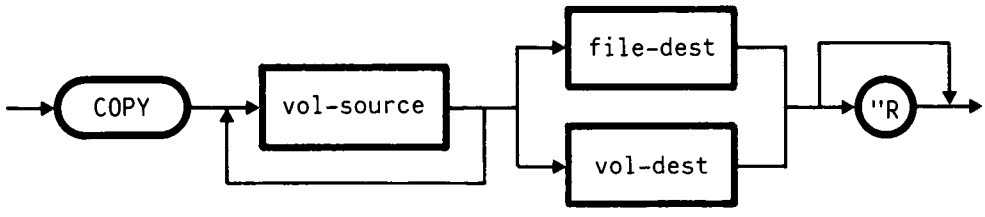
A source volume may be copied onto:

- a non-existent name

  A volume is created into which the source volume is copied.

- a destination file

  When specifying the replace option, the destination file is overwritten by the source volume, which takes the name of the destination file.



where:

vol-source is the name or path name of the source volume.

file-dest is the name or path name of the destination file.

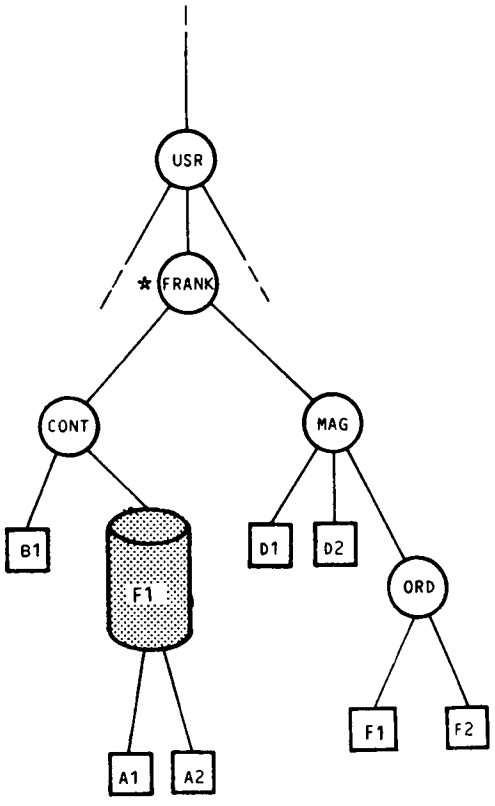
vol-dest is the name or path name of the destination volume.

"R is the replace option.

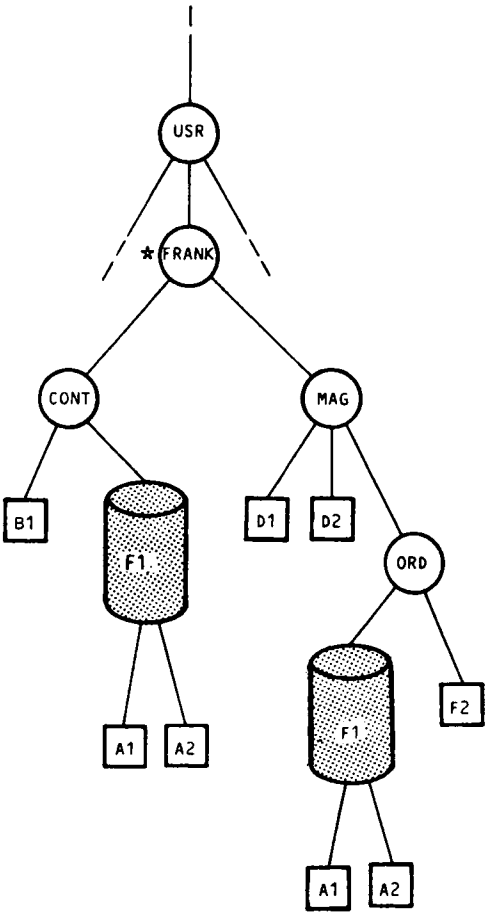
Examples

1. COPY CONT/F1 MAG/ORD "R

before

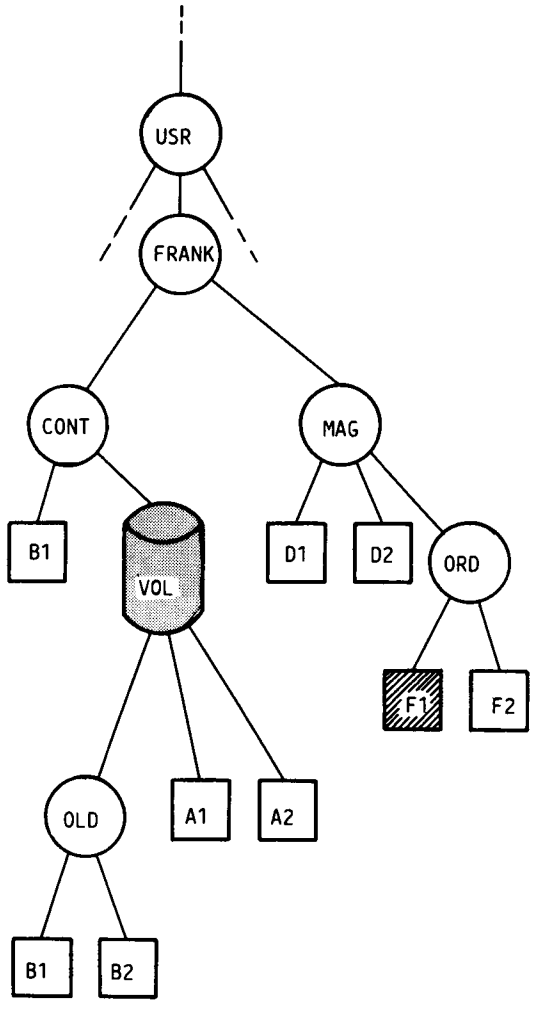


after

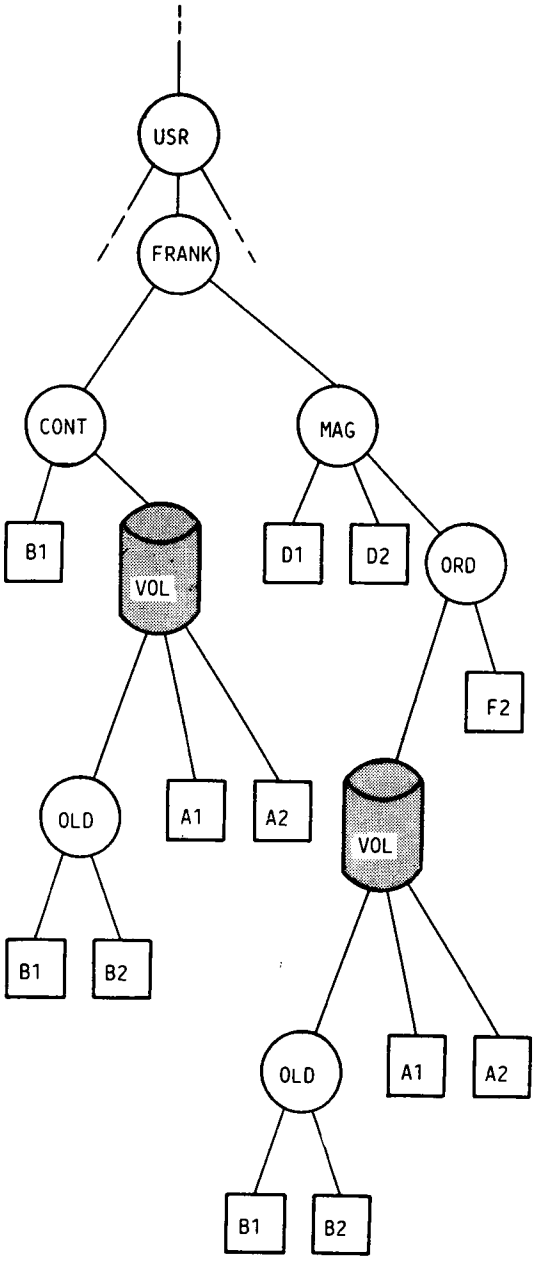


2. COPY CONT/VOL MAG/ORD/F1 "R (where F1 is a byte-stream file)

before



after



Results Tables

The following tables show the actions carried out when a copy command is executed.

source: byte-stream file
destination:

does not exist		byte-stream file		positional file		keyed file		directory		volume	
opt.	action	opt.	action	opt.	action	opt.	action	opt.	action	opt.	action
-, "R	create copy	-	error	-	error	-	error	-, "R	insert	-, "R	insert
		"R	remove create copy	"R	remove create copy	"R	remove create copy				
"A		"A	append	"A	error	"A	error	"A	insert or append	"A	insert or append

source: positional file
destination:

does not exist		byte-stream file		positional file		keyed file		directory		volume	
opt.	action	opt.	action	opt.	action	opt.	action	opt.	action	opt.	action
-	create copy	-	error	-	error	-	error	-, "R	insert	-, "R	insert
"R	create copy	"R	remove create copy	"R	remove create copy	"R	remove create copy				
"A	error	"A	error	"A	error	"A	error	"A	error	"A	error

source: keyed file
destination:

does not exist		byte-stream file		positional file		keyed file		directory		volume	
opt.	action	opt.	action	opt.	action	opt.	action	opt.	action	opt.	action
-	create copy	-	error	-	error	-	error	-, "R	insert	-, "R	insert
"R	create copy	"R	remove create copy	"R	remove create copy	"R	remove create copy				
"A	error	"A	error	"A	error	"A	error	"A	error	"A	error

source: list of files
destination:

does not exist		byte-stream file		positional file		keyed file		directory		volume	
opt.	action	opt.	action	opt.	action	opt.	action	opt.	action	opt.	action
- "R "A	error	- "R "A	error	- "R "A	error	- "R "A	error	- "R "A	insert	- "R "A	insert

source: directory
destination:

does not exist		byte-stream file		positional file		keyed file		directory		volume	
opt.	action	opt.	action	opt.	action	opt.	action	opt.	action	opt.	action
-	create copy	-	error	-	error	-	error	-		-	
"R	create copy	"R	remove create copy	"R	remove create copy	"R	remove create copy	"R	error	"R	error
"A	error	"A	error	"A	error	"A	error	"A		"A	

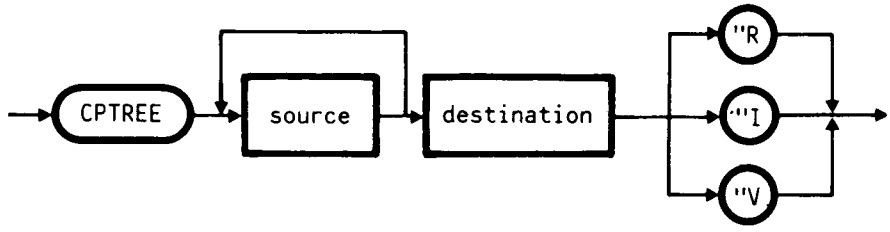
source: volume
destination:

does not exist		byte-stream file		positional file		keyed file		directory		volume	
opt.	action	opt.	action	opt.	action	opt.	action	opt.	action	opt.	action
-	create copy	-	error	-	error	-	error	-		-	
"R	create copy	"R	physical copy	"R	remove create copy	"R	remove create copy	"R	error	"R	error
"A	error	"A	append	"A	error	"A	error	"A		"A	

Recursively copies files contained in directories and volumes, including sub-directories, into another directory or existing volume.

If a directory or volume is to be copied, the files comprising the source directory or volume are merged with those contained in the destination directory or volume.

A directory cannot be copied either onto itself or onto a sub-directory.



---

where:

source is the name or path name of the source file, directory, or volume.

destination is the name or path name of the destination directory or volume.

"R is the replace option.

"I is the option for interactive execution of the command.

"V is the option with which to display the name of the files before they are copied (verbose).

Characteristics

1. If there are file system objects in the destination directory or volume with the same name as the source files, directories, or volumes, these will only be overwritten if the replace option has been specified. Otherwise the command aborts and an error message is displayed. A directory with the same name as a source name cannot be overwritten.

2. If the option for interactive execution of the command is specified, the user is asked if the element is to be replaced or not each time a destination name is the same as a source one.

3. The command CPTREE is used instead of the COPY command when the contents of one directory are to be appended to the contents of another directory, without copying the root.

4. A program directory copy operation includes its root.

5. To copy a program directory residing in a directory into another directory already containing a directory of the same name, that directory must first be removed before copying the source program directory.

6. As execution of the CPTREE command using the replace option involves remove operations, see the REMOVE command.

7. If the interactive mode option has been specified, the following message is displayed before each file is copied:

   O.K. TO COPY file name (Y/N):

   the user may enter **Y** or **y** if the file is to be copied, **N** or **n** otherwise.

8. All the errors that occur during execution of CPTREE are displayed as follows:

   COPYING path name: error message

   where "path name" is the name or path name of the file system object that caused the error and "error message" specifies the nature of the error (see Appendix A).

9. The following access rights are required to execute the command:

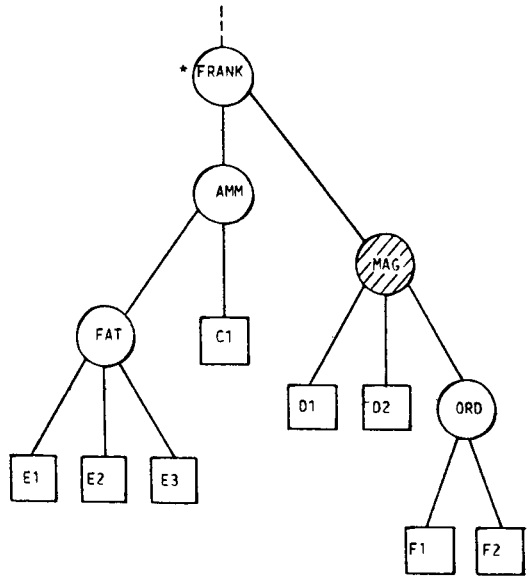
   - execution and read on the father directory of the source file system object
   - read on the file system object to copy
   - append or execution and write on the destination directory or volume.

10. The access rights defined in the default list of the user performing the operation are assigned to the file system objects copied.

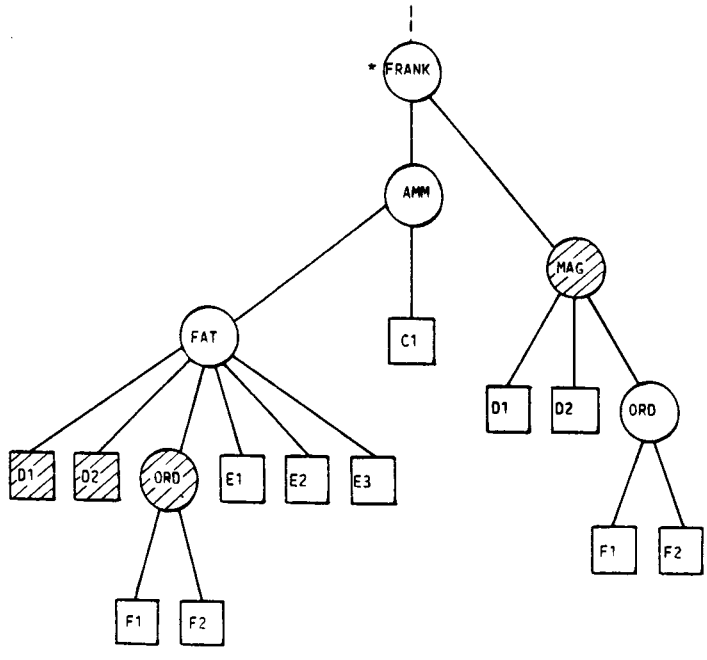
Example

CPTREE MAG AMM/FAT

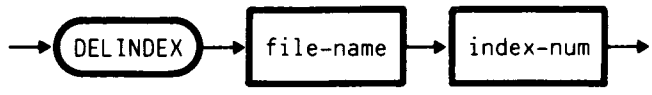
before



after



DELINDEX deletes the specified index of a keyed file. (It is essential that one index remains in existence if keyed access to the file is to required again).



where:

file-name is the name or path name of the file (local only).

index-num is the index number identifying the index (see characteristics below).

Characteristics

1. Either the primary or one of the secondary indices of a keyed file can be deleted.

2. If the primary index of a keyed file is deleted using DELINDEX, the file may no longer be written to.

3. The index of a keyed file is identified by an "index number" assigned by the system at file-creation time. This can be displayed using the PRY command on the file.

4. See also the commands: PRY, MKINDEX, REMOVE (note 2).

5. Access rights for execution and write on the father directory are required for the execution of this command. Its use is reserved to the primary owner of the file and the system administrator.

Example

In order to cancel the secondary index of the CLIENTS file whose index number is 3, the following command may be entered:

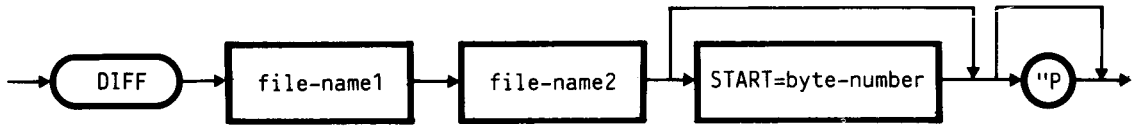
```
DELINDEX STORE/CLIENTS 3
```

Performs a byte per byte comparison of two specified byte-stream files or of the contents of two physical devices.

If the specified files are of different lengths, the comparison is performed from the beginning of each file until the end of the shortest file is reached.

If a difference is found between the two files, this is indicated by an appropriate message. It is then possible to restart the comparison from any byte position within the bounds of the shorter file. See the operator interface below.

The output of DIFF can be redirected to a printer.



where:

file-name1 is the name or path name of the first file or device.

file-name2 is the name or path name of the second file or device.

byte-number is the position of the first byte from which to start the comparison; if not specified, the value 0 is assumed by default.

"P is the continuous output option, with no interaction.

The DIFF command returns a completion code in the MCL variable %STATUS, which may have the following values:

VALUE	MEANING
0	The two files are equivalent, even if one file is larger than the other; in this case, the larger file contains all the information (byte per byte) of the smaller one.
1	Differences exist between the two files.
2	Command aborted.

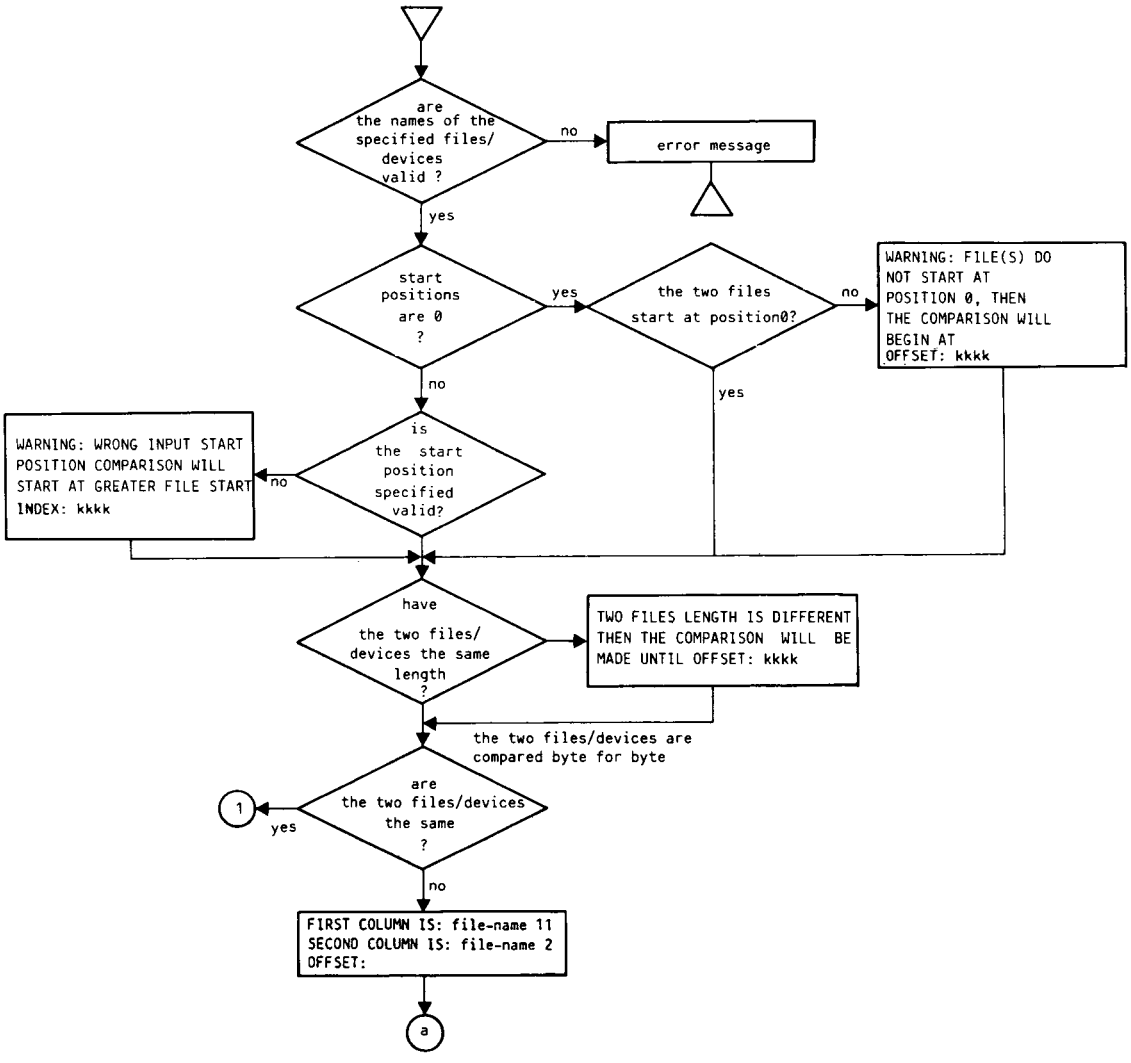


Fig. 3.DIFF-1 DIFF Command - Operator Interface (cont.)

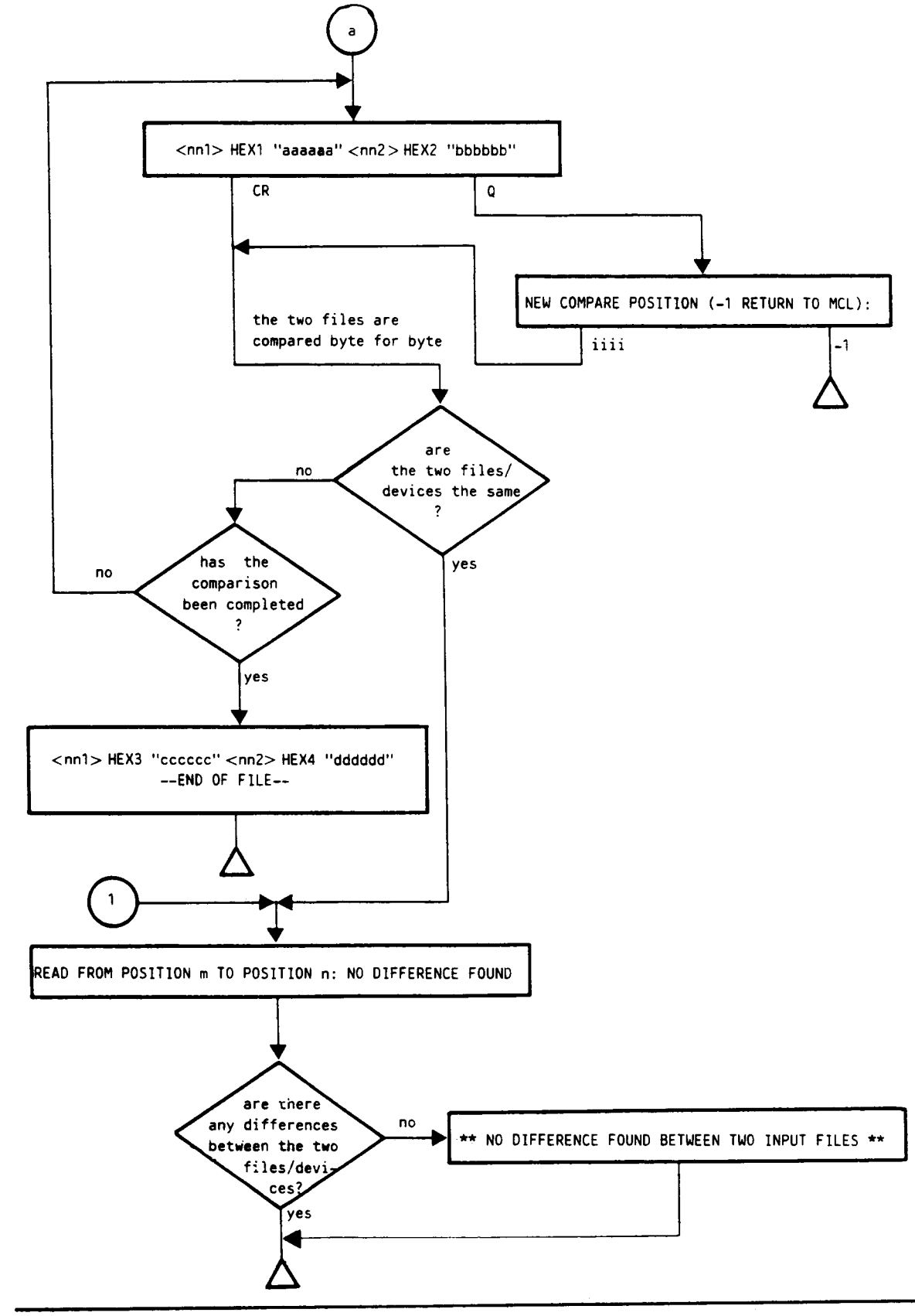


Fig. 3.DIFF-2 DIFF Command - Operator Interface

WARNING: FILE(S) DO NOT START AT POSITION 0, THEN THE COMPARISON WILL BEGIN AT OFFSET: kkkk

The initial position of the file or files is not 0, so the comparison will begin from the byte whose offset is shown as "kkkk."

WARNING: WRONG INPUT START POSITION, COMPARISON WILL START AT GREATER FILE START INDEX: kkkk

The two files start at different positions. The comparison will start from the byte whose offset is shown as "kkkk", which corresponds to the initial byte offset of the file having the greater initial byte offset.

TWO FILES LENGTH IS DIFFERENT, THEN THE COMPARISON WILL BE MADE UNTIL OFFSET: kkkk

The two files are of different length. The comparison will be made from the beginning of the files until the byte with offset "kkkk" , which is the length of the shorter file.

FIRST COLUMN IS: file-name1
SECOND COLUMN IS: file-name2
OFFSET:
<nn1> HEX1 "aaaaaa" <nn2> HEX2 "bbbbbb"

file-name1 is the name of the file displayed in the first column.

file-name2 is the name of the file displayed in the second column.

nn1 is the offset from the first of the six bytes displayed in the first column; this is the first byte found to be different in the two files.

HEX1 is the hexadecimal representation of the six bytes from the offset "nn1" in "file-name1".

aaaaaa is the ASCII representation of the bytes shown in "HEX1".

nn2 is the offset of the first of the six bytes displayed in the second column;

HEX2 is the hexadecimal representation of the six bytes from the offset "nn2" in "file-name2".

bbbbbb is the ASCII representation of the bytes shown in "HEX2".

CR The comparison of the two files continues from the byte with offset "nn1" + 6 for the first file and "nn2" + 6 for the second.

Q The comparison is halted.

NEW COMPARE POSITION (-1 RETURN TO MCL) :

This appears after the comparison is halted. The user may continue the comparison from a new offset or return to the Shell environment.

iiii This represents the offset from which the comparison is to continue, and must be a number smaller than the size of the smallest of the two files.

-1 Exits from the command DIFF and returns to Shell.

<nn1> HEX3 "cccccc" <nn2> HEX4 "dddddd"
-- END OF FILE --

This shows the last six bytes which may be compared by DIFF. In this and the preceding message, the characters "b" or "-" may appear in the hexadecimal or ASCII sections. These characters are added by DIFF to complete the display of six bytes when an unprintable character code is encountered.

READ FROM POSITION m TO POSITION n: NO DIFFERENCE FOUND

During the comparison, no differences were found between the two files from byte "m" to offset "n".

** NO DIFFERENCE FOUND BETWEEN TWO INPUT FILES **

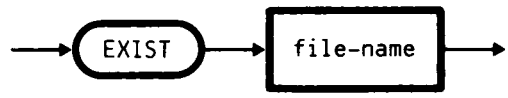
The two input files are the same.

Characteristics

1. To execute this command, the following access rights are required: execution on the father directory or directories, and read on the files to be compared.

2. See Appendix A for the error messages.

Checks for the existence of a specified file.



where:

file-name is the file name or path name.

The result of this check is loaded in the %STATUS variable which can assume the following values:

VALUE	MEANING
0	The file exists (any type).
1	The name exists but does not correspond to a file (e.g. volume or directory name).
2	The name is non-existent.

Characteristics

Access rights to execute and read the father directory are required to perform this command.

Example

```
"TESTFOR" EXAMPLE
RDNEXT %FILE;
EXIST %FILE;
IF %STATUS=0
  THEN ECHO 'EXISTS'
  ELSE ECHO 'DOESN''T EXIST';
```

FCU (Floppy Conversion Utility) can convert files from ASCII to EBCDIC or vice versa. The source and destination discs may be either 8" or 5", but they must be written to the format standards common to L1, IBM, and many other systems (that is ECMA 58 and 67, or Olivetti Standard 17).

The utility is primarily intended to interface between Olivetti and IBM compatible systems (that is, between ASCII and EBCDIC formats), but 5" floppies produced by FCU are also acceptable to the Olivetti M20, through the use of its conversion program.

FCU FUNCTIONS

FCU can:

- perform file conversion to L1 format (that is, from ECMA standard EBCDIC to Olivetti standard ASCII), on another disc
- perform file conversion to "external" format (that is, from Olivetti standard ASCII to ECMA standard EBCDIC), on another disc
- display the floppy volume header label
- display the floppy file header label.

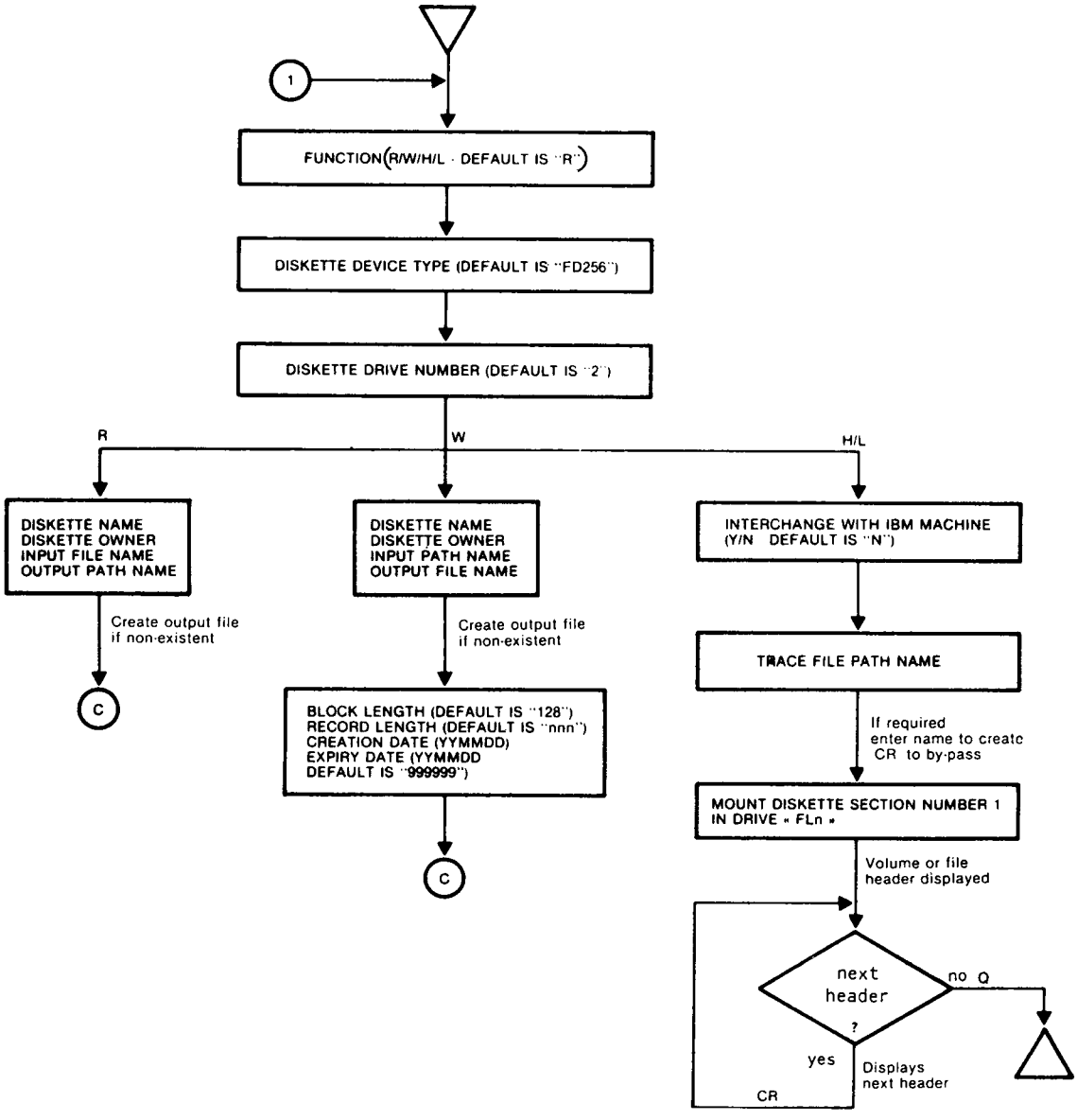


Fig. 3.FCU-1 FCU Command - Operator Interface (cont.)

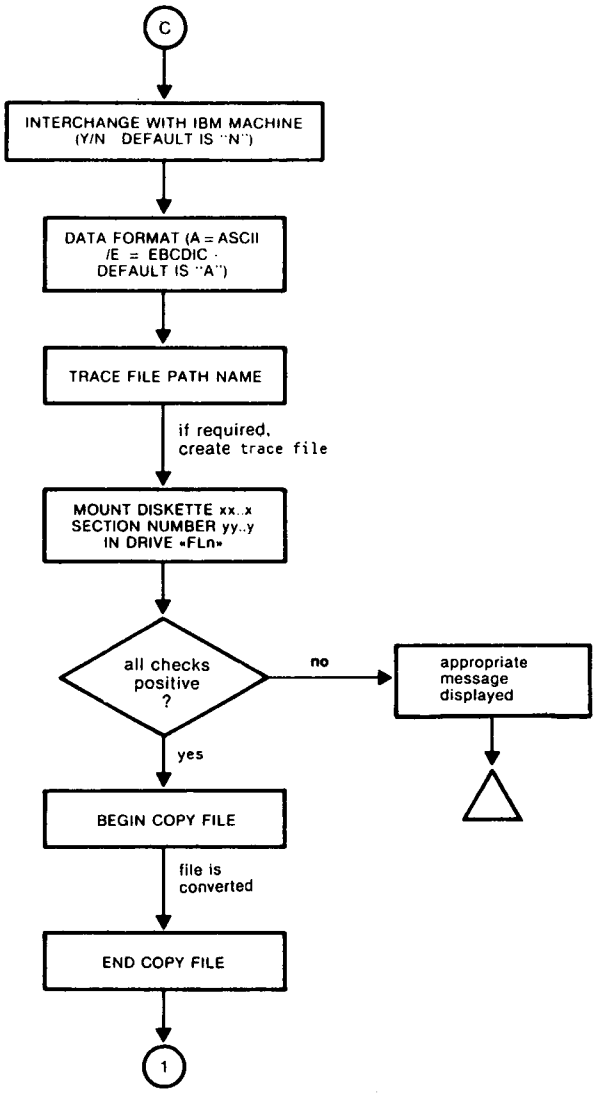


Fig. 3.FCU-2 FCU Command - Operator Interface

FUNCTION (R/W/H/L - DEFAULT IS 'R')

The user chooses the required function:

R	Selects READFLOP, to convert data from non Olivetti format (EBCDIC) into L1 format (ASCII), also converting the volume header label.
W	Selects WRITEFLOP, to convert data from L1 format (ASCII) into non Olivetti format (EBCDIC), also converting the volume header label.
H	Displays the volume header label.
L	Displays the first file header label.
CR	If entered on its own, selects the default value READFLOP.

DISKETTE DEVICE TYPE (DEFAULT IS 'FD256')

xx..x	Specifies the type of floppy disc to be read, that is one of: FD256, FD1MB, MF1MB or MF320.
CR	If entered on its own, selects the default value FD256.

DISKETTE DRIVE NUMBER (DEFAULT IS '2')

n	Identifies the drive where the input floppy is loaded.
CR	If entered on its own, selects the default drive value 2.

For both READFLOP and WRITEFLOP

DISKETTE NAME

xx..x Enter the volume name of the floppy (maximum of 6 alphanumeric characters):

- If converting to L1 format, the name entered must be the name already present on the input floppy disc, for checking purposes.
- If converting from L1 format, this name is written on the output floppy disc.

DISKETTE OWNER

xx..x Enter the name of the owner of the output floppy (maximum of 14 alphanumeric characters):

- If converting to L1 format, the name entered must be the name already present on the input floppy disc, for checking purposes.
- If converting from L1 format, this name is written on the output floppy disc.

ACTION (R = RETRY, A = ABORT - DEFAULT IS 'A')

The requested action has failed, (due to incompatible input or an error condition), and the user is presented with a choice:

A to abort the function (READFLOP or WRITEFLOP), and return to Shell

R to return to the prompt immediately preceding the failed action

CR to select the default action, which is to abort and return to Shell.

For READFLOP only

INPUT FILE NAME

xx..x Enter the name of the file to be read from the floppy disc.

OUTPUT PATH NAME

xx..x Enter the path name of the file which is to receive the converted data. If the file does not yet exist, it is created by the utility.

For WRITEFLOP only

INPUT PATH NAME

xx..x Enter the path name of the input file (source).

OUTPUT FILE NAME

xx..x Enter the output (destination) file name (maximum of 6 alphanumeric characters). If the file does not already exist, the utility creates it.

BLOCK LENGTH (DEFAULT IS '128')

nn...n Specify the requested block length of the output file, in bytes (the block length must not be greater than the track size of the output disc). The default value is 128 bytes.

RECORD LENGTH (DEFAULT IS 'nnn')

where the default value "nnn" is:

128 if the output device type is FD256,
256 if the output device type is FD1MB, MF1MB, or MF320.

nn...n Indicate the required record length, which must not be greater than the stated default. This represents the byte length of a sector on the output floppy disc.

CR Entered on its own, selects the default record length.

CREATION DATE (YYMMDD)

yymmdd Specify the creation date of the output (converted) file. In the format shown, "yy" represents the year, "mm" the month, and "dd" the day (for example, 860102 represents the 2nd January 1986).

EXPIRY DATE (YYMMDD - DEFAULT IS "999999")

yymmdd Specify the date on which the converted data is expected to become obsolete. The format is the same as described just above.

CR Entered on its own, selects the default "999999" (which may be used to ensure that the data will not be overwritten).

INTERCHANGE WITH IBM MACHINE (Y/N - DEFAULT IS 'N')

Specify whether the data exchange is with an IBM machine:

Y The "exchange" floppy disc comes from or is intended for an IBM machine, and must have an IBM-type volume header label (type "W").

N The "exchange" floppy disc comes from or is intended for a non-IBM machine, and must have an ECMA standard label (type "3").

CR Entered on its own, selects the default **N** response.

DATA FORMAT (A = ASCII / E = EBCDIC - DEFAULT IS 'A')

Specify the conversion to be performed, which in turn defines the conversion table to be used. This may be a user defined conversion table:

A for conversion to ASCII format

E for conversion to EBCDIC format

CR if entered on its own, selects the default conversion to ASCII format.

TRACE FILE PATH NAME

xx..x Specify the path name of a trace file to be written with details of the conversion operations performed. If the file already exists, this trace information is appended to it. If the file does not yet exist, it is created here by FCU.

CR Entered on its own, bypasses the trace option.

MOUNT DISKETTE xx..x SECTION NUMBER yy..y IN DRIVE <FLn>

Mount the floppy disc with volume name "xx..x" and section number "yy..y" in drive "FLn" or "MFn":

CR Indicates the end of the mount operation.

Display Functions H (Volume Header) or L (File Header)

INTERCHANGE WITH IBM MACHINE (Y/N - DEFAULT IS "N")

If the disc produced is to be read by an IBM machine, answer **Y.**

TRACE FILE PATH NAME

If a trace file is required by the user, enter here the path name intended for it, and the file is created. Otherwise, press **CR** to bypass.

MOUNT DISKETTE SECTION NUMBER 1 IN DRIVE <FLn>

This is an instruction or reminder to mount the first disc.

TRACE FILE

Users can also record the details of the conversion(s) in a trace file, which will remain intact until the user specifies the same path name for another use.

This is particularly useful when several conversion operations are performed, because this file can then be listed or printed, using normal Shell commands.

The information contained in a trace file appears thus:

```
CONVERT UTILITY PARAMETERS:
    FUNCTION                        = READFLOP
    DISKETTE DEVICE TYPE            = MF320
    DISKETTE NAME                   = M20DISK
    DISKETTE OWNER                  = OLIVETTI
    INPUT FILE NAME                 = DATAFILE
    OUTPUT FILE NAME                = M20DATA
    BLOCK LENGTH                    = 256
    RECORD LENGTH                   = 256
    INTERCHANGE WITH IBM MACHINE    = YES
    DATA FORMAT                     = EBCDIC
***  RESULT  = CORRECT TERMINATION    ***
```

DISPLAY OF THE HEADERS

The volume header information is displayed thus:

```
LABEL IDENTIFIER     -  Label referred to (obviously "VOL").
VOLUME IDENTIFIER    -  Volume name, as found in the label.
ACCESSIBILITY INDIC. -
OWNER IDENTIFIER     -  Name of the owner.
FORMAT INDICATOR     -
PHYS. REC LEN. IDEN  -
LABEL TYPE           -
```

The file header information is displayed thus:

LABEL IDENTIFIER	- Label referred to (obviously HDR1).
FILE IDENTIFIER	- File name, as found in the label.
BLOCK LENGTH	- Bytes per block.
BEGIN OF EXTENT	-
END OF EXTENT	-
RECORD FORMAT	-
BYPASS INDICATOR	-
ACCESSIBILITY INDICATOR	-
WRITE PROTECT	-
INTERCHANGE TYPE IND.	-
MULTIVOLUME INDICATOR	-
FILE SECTION NUMBER	-
CREATION DATE	- In the format "YYMMDD".
RECORD LENGTH	- Bytes per record.
OFFSET TO NEXT RECORD SP	-
RECORD ATTRIBUTE	-
FILE ORGANISATION	-
EXPIRATION DATE	- Either a date in format "YYMMDD", or "999999" to indicate no expiry.
VERIFY/COPY INDICATOR	-
END OF DATA	-
TO CONTINUE PRESS (CR) OR (Q) TO EXIT	- Allows the header of the next file on disc to be displayed.

Information Messages

>> BEGIN COPY FILE

Conversion has begun.

>> END COPY FILE

Conversion has ended.

Characteristics

1. FCU is usually used interactively, but it can also be used in batch mode if the user edits an "answers" file, then redirects it to the standard input.

2. FCU will read or write only in ASCII or EBCDIC codes.

3. The only type of file that FCU can convert for another system is byte-stream; program files, packed data files, or object files cannot be converted.

4. FCU must be configured as a program directory, in which the command FCU is renamed as "MAIN", and the alias file name "FCUTABLE" points to the FCUTABLE file containing the two conversion tables ASCII and EBCDIC.

   The user may construct his own tables using FCUTAB. The alias file name FCUTABLE must then point to those instead.

5. An "exchange" disc written by FCU for another system is allowed to contain only one file, but a very large file may be continued onto several discs.

6. If there are several files of the same name on a disc, only the first occurrence can be accessed by FCU.

7. If required, a trace file can be created and eventually listed and printed, showing details of the conversion.

8. As usual, the volume header label, containing information about the disc and the files on it, is found on the outermost cylinder of the disc, cylinder 0. READFLOP expects to find this information on cylinder 0, and WRITEFLOP writes it. For more details, see Appendix D.

9. For the conversion from EBCDIC to ASCII, access rights for execution and append on the directory containing the file in format L1 are required.

   The converse operation, from ASCII to EBCDIC, requires the following rights:

   - read on the file in format L1

   - execution and read on the directory containing the file in format L1.

>> xx...x FILE NOT FOUND

The specified file does not exist.

>> DATE ERROR

At least one of the characters entered for "yymmdd" (year, month, day) is invalid.

>> PROGRAM ABORTED

When executing the conversion utility an error occurred which does not permit normal termination.

>> THERE ARE CHARACTERS NOT PERMITTED IN INPUT STRING

There are illegal characters in the input string.

>> VALUE ERROR

The character string entered is in the wrong format, or contains a non-alphanumeric character.

>> VALUE TOO LONG

The character string entered is too long.

>> VALUE TOO SMALL

The character string entered is too small.

DISKETTE NAME NOT EQUAL TO ENTERED NAME

The name of the floppy disc inserted is different from that entered by the user.

DISKETTE OWNER NOT EQUAL TO ENTERED NAME

The owner name of the floppy disc inserted is different from that entered by the user.

DISKETTE SECTION NUMBER WRONG

The file section number of the floppy disc inserted is different from that entered by the user.

ERROR READING VOLUME HEADER

An error occurred while reading the volume header label.

RECORD LENGTH MUST BE LESS THAN OR EQUAL TO BLOCK LENGTH

The record length must not be greater than the disc block size.

VOLUME IS NOT CORRECT

The volume name is incorrect.

WRONG DISKETTE

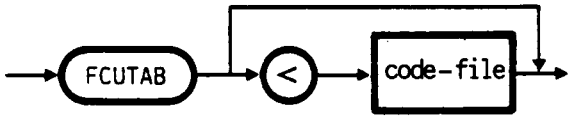
The type of the floppy disc inserted is not supported by the system.

Creates a user-defined conversion table for FCU.

FCUTAB overwrites the data stored in the file called FCUTABLE. This file is pointed to by the alias file FCUTABLE in the program directory FCU.

The user must save a copy of FCUTABLE before using FCUTAB if the original ASCII-EBCDIC conversion table supplied with FCU must be retained.

When activated, FCUTAB requests the hexadecimal values of the ASCII/EBCDIC characters to substitute to the current table values. Once the table entered, FCUTAB allows the user to either enter another table or to return to the MCL prompt.



The parameters are entered interactively.

It is recommended that a file containing the required characters be generated using the system editor, and then used as redirected input to FCUTAB, for example:

FCUTAB < code-file

where:

code-file is a byte-stream file containing the required character codes for the new conversion table, each on a separate line.

Characteristics

1. The conversion table has also been extended to 256 characters for the ASCII code. In this way it may support special characters (e.g. KANA). Therefore, the user must provide, during an interactive work session, the 256 characters for both codes. If no conversion is required, the characters "FF" must be entered.

2. Required access right on the FCUTABLE file are in reading so that it can be read by FCU, and in writing so that it can be updated by FCUTAB.

ASCII

$b_4 b_3 b_2 b_1$	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0 0 0 0	NUL	TC_7 (DLE)	SP	0	@	P	`	p								
0 0 0 1	TC_1 (SOH)	DC_1	!	1	A	Q	a	q								
0 0 1 0	TC_2 (STX)	DC_2	"	2	B	R	b	r								
0 0 1 1	TC_3 (ETX)	DC_3	#	3	C	S	c	s								
0 1 0 0	TC_4 (EOT)	DC_4	¤	4	D	T	d	t								
0 1 0 1	TC_5 (ENQ)	TC_8 (NAK)	%	5	E	U	e	u								
0 1 1 0	TC_6 (ACK)	TC_9 (SYN)	&	6	F	V	f	v								
0 1 1 1	BEL	TC_{10} (ETB)	'	7	G	W	g	w								
1 0 0 0	FE_0 (BS)	CAN	(	8	H	X	h	x								
1 0 0 1	FE_1 (HT)	EM	)	9	I	Y	i	y								
1 0 1 0	FE_2 (LF)	SUB	*	:	J	Z	j	z								
1 0 1 1	FE_3 (VT)	ESC	+	;	K	[	k	{								
1 1 0 0	FE_4 (FF)	IS_4 (FS)	,	<	L	\	l	|								
1 1 0 1	FE_5 (CR)	IS_3 (GS)	-	=	M	]	m	}								
1 1 1 0	SO	IS_2 (RS)	.	>	N	^	n	—								
1 1 1 1	SI	IS_1 (US)	/	?	O	_	o	DEL								

EBCDIC

$b_4 b_3 b_2 b_1$	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0 0 0 0	NUL	DLE			SP	&	-						{	}	\	0
0 0 0 1	SOH	DC_1					/		a	j	-		A	J		1
0 0 1 0	STX	DC_2		SYN					b	k	s		B	K	S	2
0 0 1 1	ETX	DC_3							c	l	t		C	L	T	3
0 1 0 0									d	m	u		D	M	U	4
0 1 0 1	HT		LF						e	n	v		E	N	V	5
0 1 1 0		BS	ETB						f	o	w		F	O	W	6
0 1 1 1	DEL		ESC	EOT					g	p	x		G	P	X	7
1 0 0 0		CAN							h	q	y		H	Q	Y	8
1 0 0 1		EM						`	i	r	z		I	R	Z	9
1 0 1 0					[	]	|	:								
1 0 1 1	VT			.	.	¤	,	#								
1 1 0 0	FF	FS		DC_4	<	*	%	@								
1 1 0 1	CR	GS	ENQ	NAK	(	)	_	'								
1 1 1 0	SO	RS	ACK		+	;	>	=								
1 1 1 1	SI	US	BEL	SUB	!	^	?	"								

Information Messages

DO YOU WANT TO INSERT ASCII TO EBCDIC TABLE (Y/N):

Y Selects the ASCII to EBCDIC table for redefinition.

N Proceeds with the next question.

DO YOU WANT TO INSERT EBCDIC TO ASCII TABLE (Y/N):

Y Selects the EBCDIC TO ASCII table for redefinition.

N Returns control to Shell.

TYPE EBCDIC HEXADECIMAL CODE TO CONVERT 'nn' ASCII CODE:

Enter the hexadecimal code of the EBCDIC character which is to be used in place of the ASCII character 'nn'.

TYPE ASCII HEXADECIMAL CODE TO CONVERT 'nn' EBCDIC CODE:

Enter the hexadecimal code of the ASCII character which is to be used in place of the EBCDIC character 'nn'.

Error Messages

> ERROR LENGTH

The value entered had more than two characters.

> VALUE ERROR

The value entered was not a valid hexadecimal value. Valid characters are 0-9, A-F.

> PROGRAM ABORTED <

The program is unable to connect the specified file.

FFT (Floppy File Transfer) is a utility which enables files to be transferred from the S6000 system to the L1 MOS system using floppy discs.

This utility has two versions. One of these runs on the S6000 system and is used to transfer files from the file system to floppy disc. The other version runs on the L1 MOS system and transfers files from floppy disc to the file system.

FILE TYPES

The file types which are supported by FFT have the following FFT file names:

- BINARY: a binary file is made up of binary data having no logical (e.g. record) structure, such as object files created by compilers or translators and the load modules created by the linker.

- TEXT: a text file contains textual data which is organised in records of variable length, such as a source program.

- SEQUENTIAL: a sequential file is made up of records having a fixed length.

- RELATIVE: a relative file is made up of records having a fixed length.

- INDEXED: an indexed file is made up of records of fixed length which may be distinguished by a primary key.

The file types supported by the S6000 and L1 MOS do not have the same type names. The file being converted must first be converted to an FFT file type before it may be transferred to the other system.

The following file conversion table indicates the FFT file type to be declared for each S6000 file which is to be converted to the L1 MOS system.

S6000	FFT	L1 MOS
Positional	Binary	Byte-stream
Positional	Text	Byte-stream
Positional	Sequential	Positional with no record deletion
Keyed	Relative	Positional with record deletion
Keyed	Indexed	Keyed

Characteristics and Restrictions

1. A file whose name corresponds to an existing file on the L1 MOS file system may not be transferred. For the successful transfer of this file the existing file must first be deleted.

2. 1Mbyte floppy discs must be used to transfer files.

3. FFT cannot handle multi-disc volumes. A user wishing to copy more than one file or a file greater or equal to 1 Mbyte is advised not to fill any one disc completely.

CALLING FFT

The FFT utility is called from the Shell environment by simply entering its name:

FFT call on S6000:

```
HH:MM:SS> fft
```

FFT call on L1 MOS:

```
MCL: fft
```

FFT COMMANDS

The FFT commands are:

- DELETE
- EXIT
- HELP
- INITIALIZE
- LIST
- RESTORE
- SAVE

Only the DELETE, RESTORE and SAVE commands necessitate the use of parameters. Parameters may be entered directly after the command or in interactive mode.

Lower or uppercase letters may be used when entering these commands. It is only necessary to enter the first command letter.

The following is a brief description of each command:

DELETE

Deletes a file previously saved on floppy disc.

D[ELETE] file-name

---

where:

file-name is the name of the file to be deleted.

Note: FFT does not manage file reorganisation and compacting. The disc space recovered from the deleted file may thus be used again only when the deleted file is the last one loaded.

EXIT

Forces the execution of FFT to end. The system returns to the Shell environment. The command has no parameters.

HELP

Displays a list of all the FFT commands giving a brief description of each one.

INITIALIZE

Initialises the floppy disc to FFT standards, deleting any data which may be already stored on the disc. The command must be executed before a new disc may be used or to delete the old contents of an used disc. The command has no parameters.

LIST

Displays the name, type and size of each file stored on the floppy disc. The command has no parameters.

RESTORE

Copies a file stored on floppy disc onto the working directory of the L1 MOS user. The original file is not deleted. The command syntax is:

R[ESTORE] floppy-name [L1-name]

---

where:

floppy-name is the name of the file stored on floppy disc.

L1-name is the name with which the file is to be stored. If not specified the file will be stored with its original name.

SAVE

Copies a S6000 file onto a floppy disc. The command syntax is:

S[AVE] S6000-name [floppy-name]

---

where:

S6000-name is the name of the file to be copied.

floppy-name is the name with which the file is to be copied onto floppy disc. If omitted the file retains its original name.

Note: The user must enter the file type when the appropriate message appears on screen.

HOW TO USE THE FFT

The following table lists the sequence of operations to be carried out when copying an S6000 file onto the L1 MOS system.

STEP	ACTION ON S6000	RESULT/COMMENT
1	Insert the floppy in drive1.	In most configurations the device name is 10_0.
2	Call the utility. from Shell: HH:MM:SS> fft	The list of available commands is displayed.
3	Initialise the floppy disc via the INITIALIZE command.	This step is mandatory when the floppy disc is new or when the data it contains is to be deleted.
4	Enter the command: SAVE.	The utility requests the name of the file to be transferred and the name with which it is to be saved.
	Enter the name of the file to be transferred and its new name.	The utility requests the name of the file type.
	Enter the file type.	The file is transferred onto floppy disc.
5	Enter the command: EXIT	The system returns to the Shell environment.
6	Remove the floppy disc from drive1.	

>>

STEP	ACTION ON L1 MOS	RESULT / COMMENT
7	Insert the floppy disc on drive1 of the L1 machine.	The device name is DEV/FL1.
8	Call the utility from Shell: MCL: fft	The list of available commands is displayed.
9	Enter the command: LIST	The list of files stored on the floppy disc is displayed (this command is optional).
10	Enter the command: RESTORE	The utility requests the name of the file to be transferred and the name with which it is to be stored.
	Enter the name of the file to be transferred and its new name.	The file is stored under the working directory.
11	Enter the command: EXIT	The system returns to the Shell environment.

NOTES

The following notes may be of interest to the user. They concern:

- the floppy disc format after it has been initialised
- the transfer of files which have secondary keys.

Format of an FFT floppy disc

The floppy disc format is given in the table below:

HEADER BLOCK:

Address	Description
0	8 bytes string containing the floppy disc identifier ('FFT V1.2').
8	16 bit integer containing the header length in bytes.
10	16 bits integer containing the number of files saved.
12+48*(n-1)	32 bits integer containing the address of the nth file.
16+48*(n-1)	32 bits integer containing the length of the nth file.
20+48*(n-1)	32 bits string containing the name of the nth file (must terminate with "ii").
52+48*(n-1)	16 bits code indicating the name of the nth file.
54+48*(n-1)	16 bits integer indicating the record length in bytes.
56+48*(n-1)	16 bits integer indicating the offset of the key within the record.
58+48*(n-1)	16 bits integer indicating the length in bytes of the key within the record.

DATA BLOCKS:

ADDRESS	DESCRIPTION
4096	Data of transferred files.

where:

- The header is a memory area of 4096 bytes which may contain information on a maximum of 85 files stored on disc.

- The size of the internal buffer for the transfer of data is 4096 bytes, records must thus be less than 4096 bytes in length. The records of a positional file must be less than 4092 bytes since 4 bytes are reserved for the record key.

- Textual file data includes the "linefeed" character which separates the records.

Transferring files with secondary keys

The FFT utility does not handle files with secondary keys. These files must thus be redefined before they may be transferred to the L1 MOS system.

Example:

The debug file "file name_DB" with secondary keys is to be transferred to the L1 MOS system. The file records have the following keys:

- a primary key of 4 bytes with offset 0

- a secondary key of 30 bytes with offset 5.

The following command must be specified in order to redefine the secondary keys which lose their meaning when the file is transferred.

```
MCL:    MKINDEX file name_DB D 5 30
```

---

where:

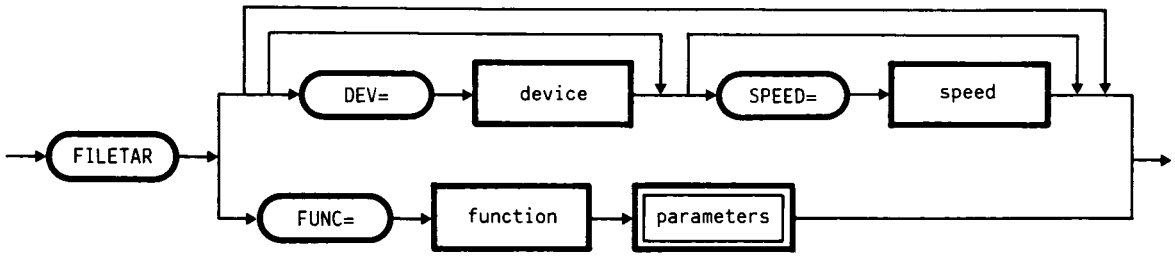
- **file name-DB** is the name of the debug file.

- **D** specifies that the secondary key may be duplicated.

- **5** indicates that the offset of the secondary key is 5 bytes.

- **30** indicates that the length of the secondary key is 30 bytes.

FILETAR allows a dump (possibly for archiving) or restore of any file system object (file, directory, or volume) between hard or floppy disc and MTU tape (with or without a file header label). It will also perform verification functions.



where:

- In non-interactive mode:

  function is one of the following functions: **S** (Save), **R** (Restore), **C** (Create), **V** (Verify), or **E** (Erase).

  parameters is the set of parameters that must be specified for each function (see further on).

- In interactive mode:

  device is the name of the tape unit used: MT1 or MT2. If the parameter is not specified, the number of the unit is requested interactively.

  speed is the tape speed, which may be 25 or 100 inches per second. If this is not specified, 25 is assumed by default.

FILETAR FUNCTIONS:

SAVE	Saves files, directories, and logical volumes from disc to tape.
RESTORE	Restores files, directories, and logical volumes from a dataset from tape to disc.
VERIFY	Checks (and displays) the header label and the name and description of every file on the tape.
CREATE	Labels/relabels a tape with a (new) volume header at its beginning (that is, "logically" erases its contents).
ERASE	Erases (physically) the contents of a tape.

Characteristics

1. FILETAR requires a "dedicated" system (that is, it takes up the entire system for itself, excluding all other work).

2. Users should not try to SUSPEND or KILL this utility, for fear of affecting the files being accessed.

3. In the interactive mode, the speed and device parameters may be specified at the same time the command is entered.

4. Files in use when a FILETAR dump is triggered are NOT saved.

5. The save function uses as many tapes as required, in sequential mode.

6. MTU tapes produced by a save operation are suitable for data exchange, as long as the unlabel option was not specified at the time the tape was created. Tapes are labelled according to the ECMA 13 standard for tape headers, making them compatible with any non-Olivetti system which observes that standard.

   FILETAR can read tapes in labelled or unlabelled form. They too must have been written to ECMA 13 standard.

   In unlabelled form, a tape only contains the volume header label, and the file identity portion of the file label.

7. The FILETAR save operation starts writing to the first available free space on tape (that is, in append mode). Thus, the same file can appear several times on one tape.

8. FILETAR can handle heterogeneous object types, that is a mixture of directories, volumes, etc.

9. If a directory is saved, all its files and sub-directories up to the last level will be saved on the tape.

10. Different operating modes can employ higher tape speeds, (see table "Tape Speed Selection Criteria" below), except for remote files, which are handled at 25 in/s maximum.

11. The MTU drive used must belong to the machine from which FILETAR is invoked, but the disc may be remote in both save and restore operations.

12. If a volume restore breaks down because of hardware failure, the part copy must be removed before restarting FILETAR. The part copy will be byte-stream in type because it represents FILETAR's work files, not yet converted to a volume.

13. The save function records the system date and time on tape, so the user must check that these are both correct before starting FILETAR.

14. More than one instance of the same file name may be found on the same tape. Each instance is identified by an occurrence number.

15. The erase function can be applied to any tape, regardless of format, block size or density. This is made possible because the function operates at a physical level and no check is done on the tape contents. It is therefore also possible to prepare for a new use tapes whose data structure and format are not compatible with FILETAR.

16. Remember that a tape can be made available quickly by deleting its contents logically, which is done by rewriting the tape header label with the creation function.

Higher tape speed availability

M30 and M40 machines can achieve a maximum tape speed of 25 in/s (inches per second), and cannot benefit from this facility.

A speed of 100 in/s is now possible on all other machines, as long as they meet certain criteria. As usual, best results depend on the user organising his save operations with an understanding of these criteria, shown below (where "STREAMING" mode corresponds to a low-level function, see Appendix C).

SPEED (inches/second)	MODE	USAGE (See "Optimisation" also)
NORMAL (25 in/s)	STREAMING	Available on all machines. Selected by user. Used for complex files, and for files <= 100K.
HIGH (100 in/s)	STREAMING	Not available on M30 or M40, but on all others having SMD or WREN hard disc. Selected by user. Used for simple files, and for files > 100K.

Tab. 1 Tape Speed Selection Criteria

Optimisation

- Save operations which include several objects avoid unnecessary tape rewinding between operations.

  However, do ensure that "fast" objects are dumped first. As soon as it has to handle a "slower" object, FILETAR goes into START/STOP mode, remaining then in that mode even if subsequent objects could be handled at a higher speed.

  Complex structures, such as directories, can be saved at 25 in/s, rather than in START/STOP mode, so the user should specify **SPEED=25** for such operations.

- Restore operations benefit from streaming mode speed if large files are restored before small ones, irrespective of their position on the tape.

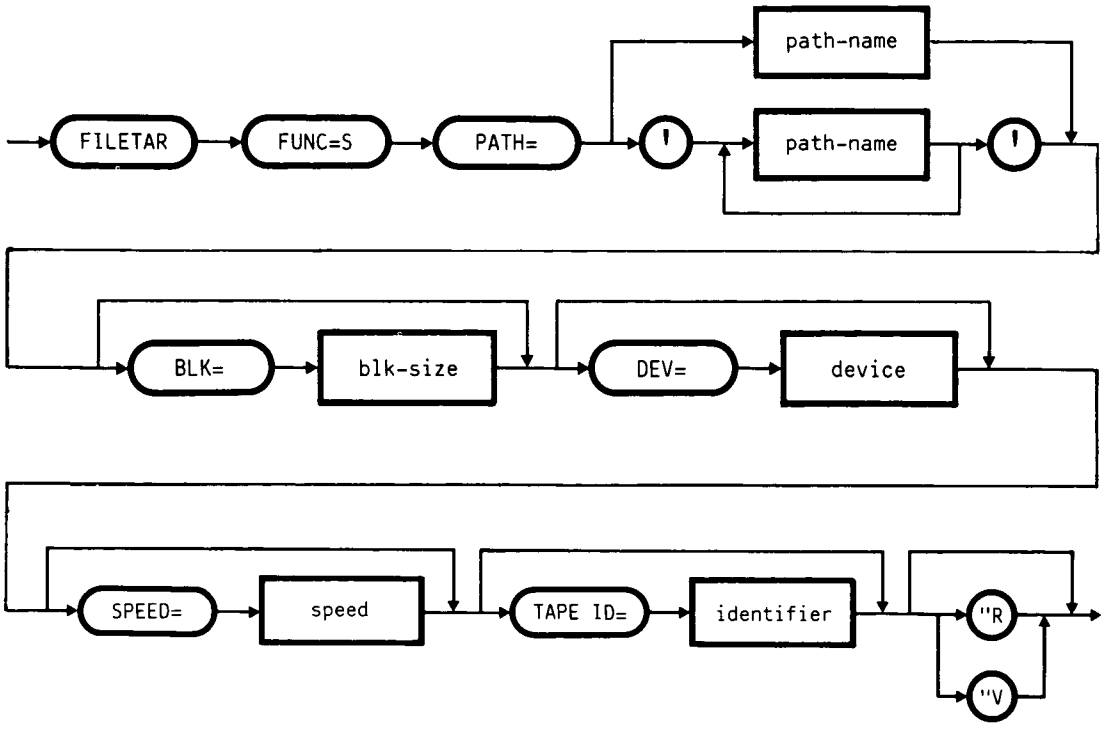
- The unlabel option significantly speeds up tape operation, especially when handling complex files.

NON-INTERACTIVE FILETAR

In non-interactive mode, each function once initiated is carried out to its end without stopping.

The functions and the parameters that can be used are described below.

SAVE FUNCTION



where:

path-name is the name or path name of the file system object (local only) to save; multiple path names (maximum of 15) must be separated by a space and the list must be enclosed in quotes.

blk-size is the block size for the data to be recorded. This must be an even number, in the range 512..16384. If the parameter is not specified, 15360 is assumed as the default value.

device is the name of the tape unit (MT1 or MT2) onto which the specified object or objects are to be saved.

speed is the tape speed, which may be either 25 or 100 inches per second. If this is not specified, 25 is assumed by default.

identifier is the tape volume identifier. If given, it is used to check that the actual tape volume identifier coincides with that specified.

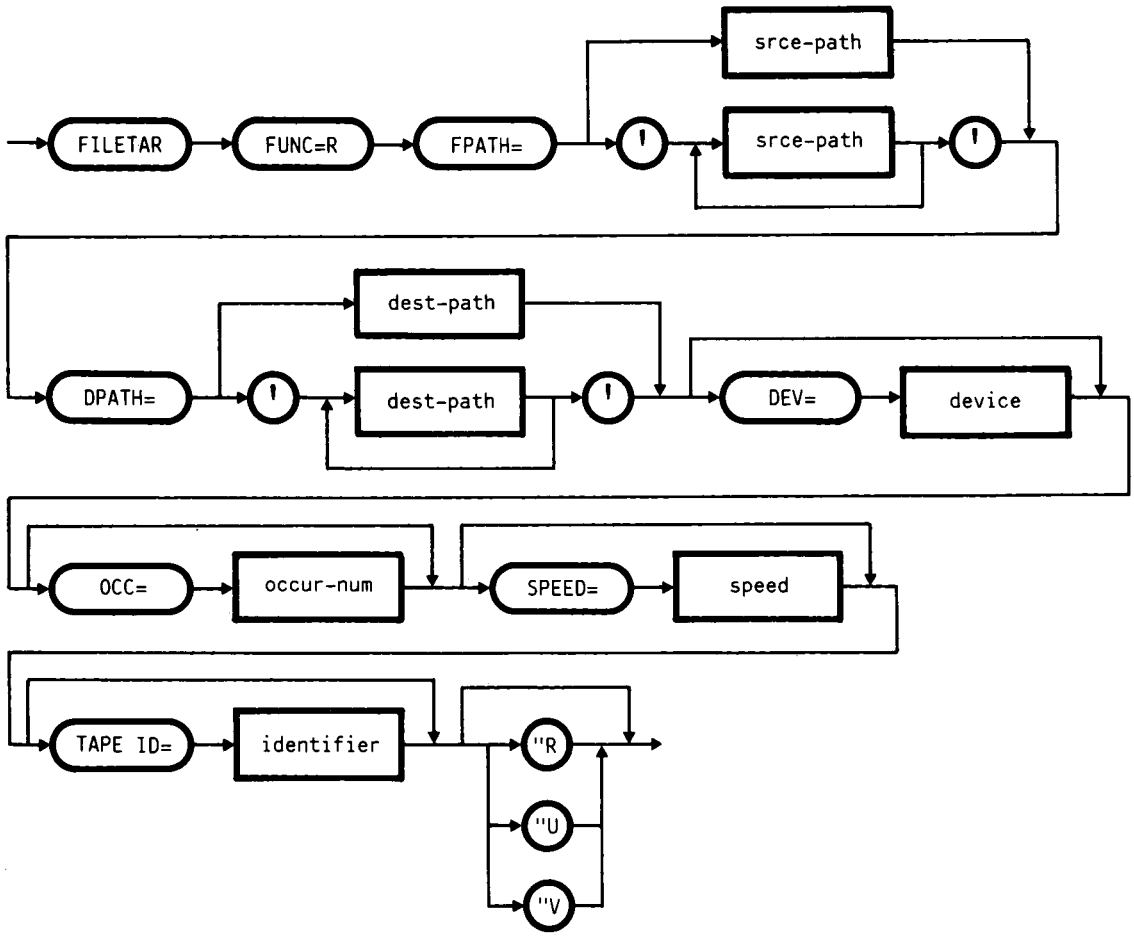
"R is the option that allows recursive saving of all the sub-directories, in the case of a saved directory.

"V traces the operations on the screen while saving is taking place.

Characteristics

If the saving operation requires several tapes, the following message is displayed each time another tape is required:

INSERT ANOTHER TAPE AND PRESS <CR>



where:

srce-path is the path name of the tape object to be restored; if there is more than one path name (maximum of 15), they must be separated by a space, and the list must be enclosed in quotes. The same path name must be used as that entered when the object is saved.

dest-path is the destination path name on disc, for the file system object to restore; if there are several path names (maximum of 15), these must be separated by a space, and the list must be enclosed in quotes.

device is the name of the tape unit (MT1 or MT2) in which the specified object or objects have been saved.

occur-num is the occurrence number of the object on the tape. If it is not specified, the first occurrence encountered is restored.

speed is the tape speed, which may be 25 or 100 inches per second. If it is not specified, 25 is assumed by default.

identifier is the tape volume identifier. If this is included it is used to check that the actual tape volume identifier coincides with the specified one.

"R specifies that any object already existing on the destination path name with the same name as that specified in "dest-path" must be overwritten.

"U allows a file to be restored using only one extent.

"V displays information about the object to restore.

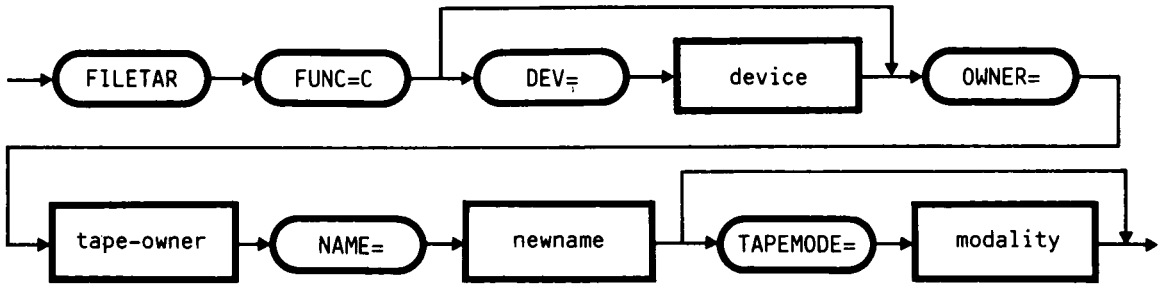
Characteristics

Usually, the access rights defined in the default access rights list are assigned to the object to be restored. The following are an exception to this:

1. If the option **"R** is specified, the access rights of the recovered object are not modified with respect to those of the existing object.

2. When a volume is being recovered, the access rights and the owner of the objects in it do not change.

CREATE FUNCTION



where:

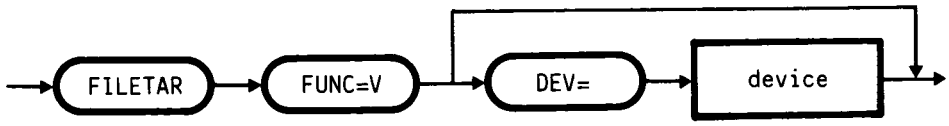
device is the name of the tape unit (MT1 or MT2).

tape-owner is the identifier of the tape owner.

newname is the identifier to assign to the new volume.

modality defines the labelling mode of the MTU tape: **U** (unlabel), or **E** (ECMA 13). If this is not specified, **U** is assumed by default.

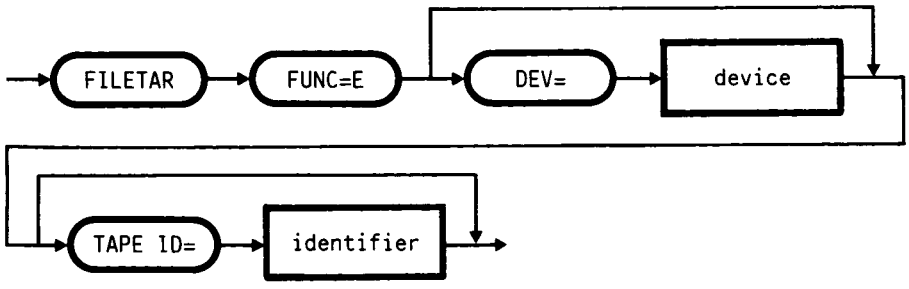
VERIFY FUNCTION



where:

device is the name of the tape unit (MT1 or MT2) to be verified.

ERASE FUNCTION



where:

device is the name of the tape unit (MT1 or MT2).

identifier is the identifier of the tape volume. If this is present, it is used to check if the actual tape volume identifier coincides with the specified one.

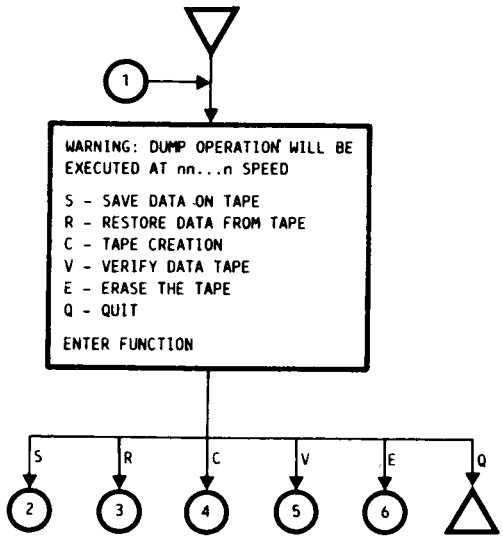


Fig. 3.FILETAR-1 FILETAR Initial Display

Information Messages

WARNING: DUMP OPERATIONS WILL BE EXECUTED AT aa...a SPEED

Displayed only the first time the main menu appears, where "aa...a" stands for either NORMAL or HIGH, the speed at which the tape will operate.

- S - SAVE DATA ON TAPE

- R - RESTORE DATA FROM TAPE

- C - TAPE CREATION

- V - VERIFY DATA TAPE

- E - ERASE THE TAPE

- Q - QUIT

- ENTER FUNCTION :

The user enters a function, completing the entry with **CR**.

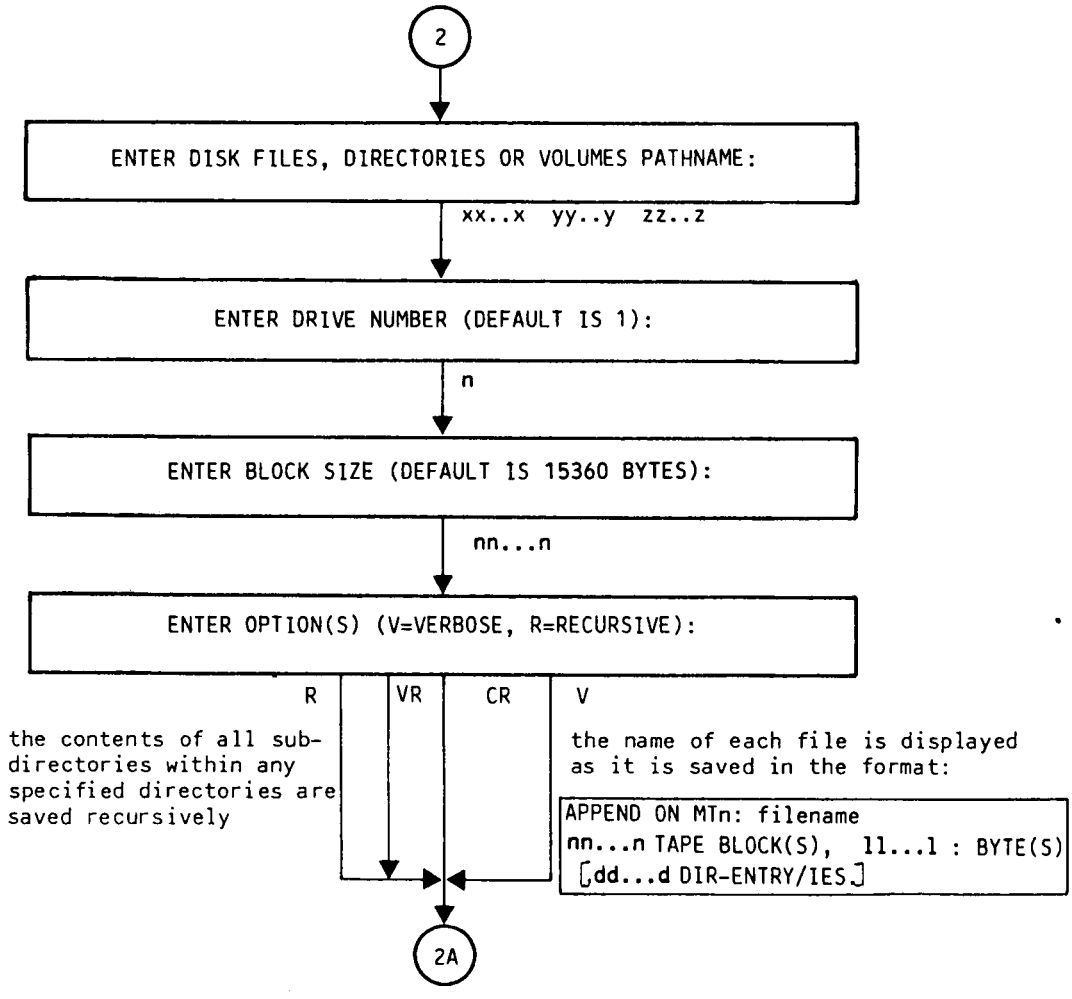


Fig. 3.FILETAR-2 FILETAR Save Command - Operator Interface (cont.)

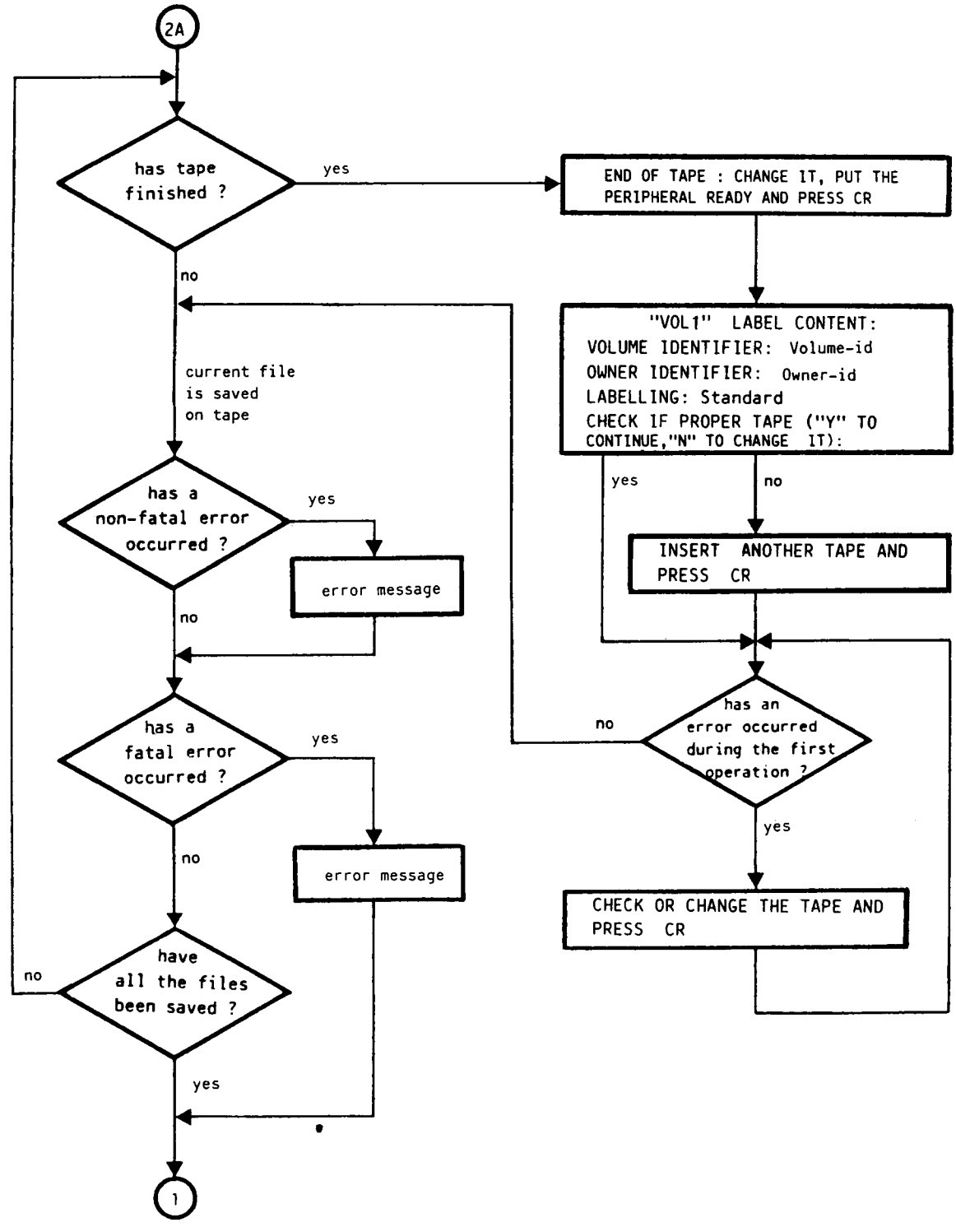


Fig. 3.FILETAR-3 FILETAR Save Command - Operator Interface

ENTER DISK FILES, DIRECTORIES OR VOLUMES PATHNAME

xx...x Enter up to 15 path names of file system objects to be saved, separated with spaces; complete the entry with **CR.** (See "Optimisation" for information on how to arrange heterogeneous files.)

In a network, the names may identify remote files (files at another site).

ENTER DRIVE NUMBER (DEFAULT IS 1):

The user enters the MTU drive number.

n Selects the MTU drive number "n".

CR Selects the MTU drive number 1 by default.

ENTER BLOCK SIZE (DEFAULT IS 15360 BYTES):

nn...n The user enters a value for the block size of the data to be recorded on the tape in the range 512..16384 bytes.

Objects bigger than 15360 bytes should be written in the optimum block size (15360 bytes).

Objects smaller than 15360 bytes should be allocated their original block size, to save space.

CR Entered on its own, selects the default block size of 15360 bytes.

ENTER OPTION(S) (V=VERBOSE/R=RECURSIVE)

V The name and size of the file system object specified is displayed in the format:

APPEND ON MTn: filename
nn...n TAPE BLOCK(S), ll...l BYTE(S) [,dd...d DIR-ENTRY/IES].

R All the "dependents" of a specified sub-directory, down to its lowest level, are saved to tape (in recursive mode) before starting on the next directory. The option **R** is mandatory if dumping a directory containing sub-directories. Otherwise, only the first level of the directory files are saved, and a warning message is displayed.

CR Entered on its own, selects neither option.

END OF TAPE: CHANGE IT, PUT THE PERIPHERAL READY AND PRESS <CR>

First, wait until the tape is rewound and unloaded automatically, in preparation for its replacement.

When the drive light stops flashing, mount another tape, set the peripheral "on-line", and press **CR**.

"VOL 1" LABEL CONTENT:
VOLUME IDENTIFIER: volume-id
OWNER IDENTIFIER: owner-id
LABELLING: standard
CHECK IF PROPER TAPE ("Y" TO CONTINUE, "N" TO CHANGE IT):

Y This requests the "create tape" function to be executed automatically on the new tape before saving files (this means a tape header label gets written at the beginning of the tape before any data is written to it).

N The user indicates that he intends to change the tape.

INSERT ANOTHER TAPE AND PRESS <CR>

Wait until the drive light stops flashing (tape is being rewound), mount another tape, set the drive "on-line", then press **CR.**

CHECK OR CHANGE THE TAPE AND PRESS <CR>

An error has occurred during the first operation on tape.

The user should verify the tape header and the condition of the tape, replacing it if necessary. When the drive light stops flashing, set the peripheral "on-line", and press **CR.**

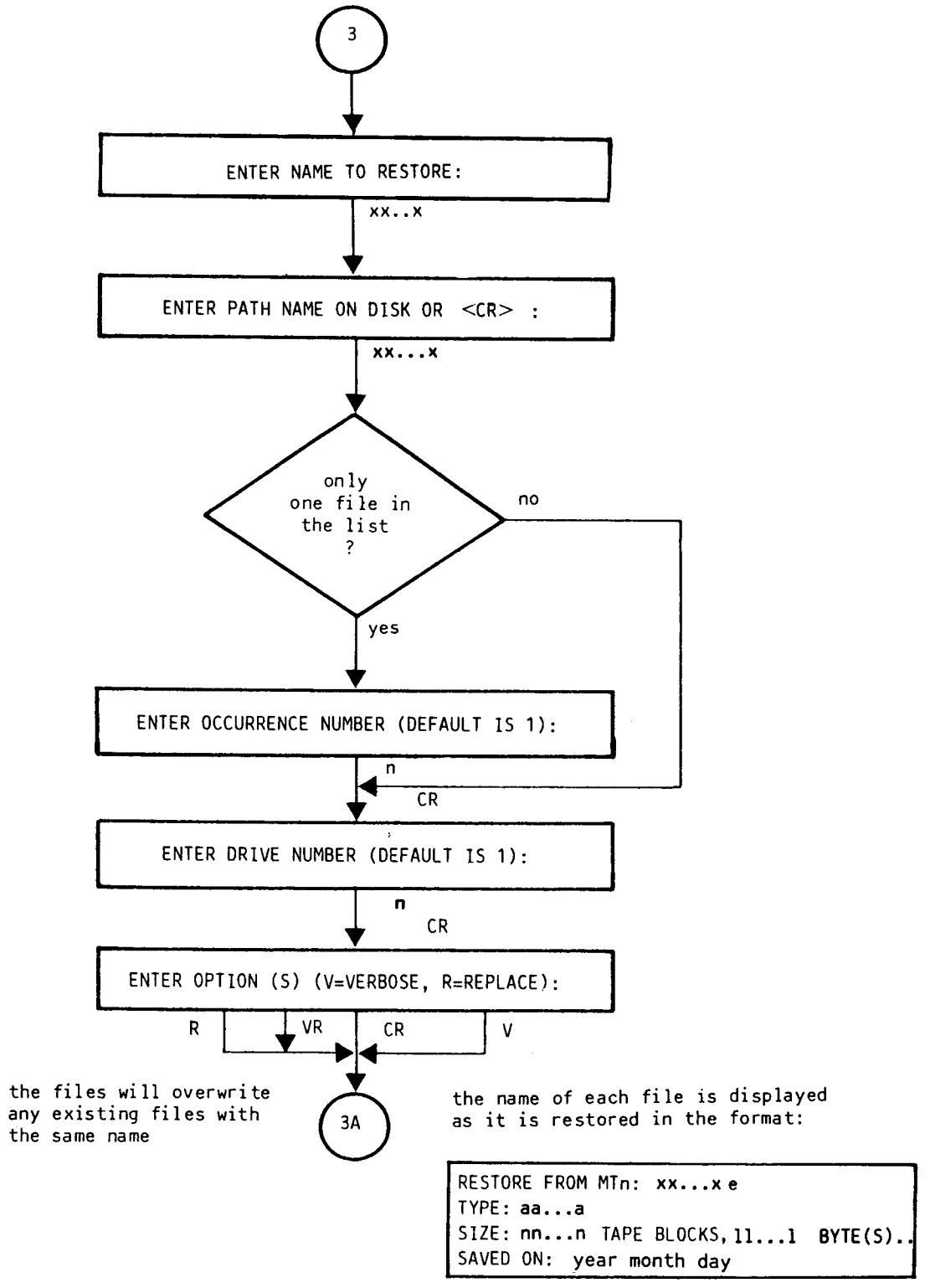


Fig. 3.FILETAR-4 FILETAR Restore Command - Operator Interface (cont.)

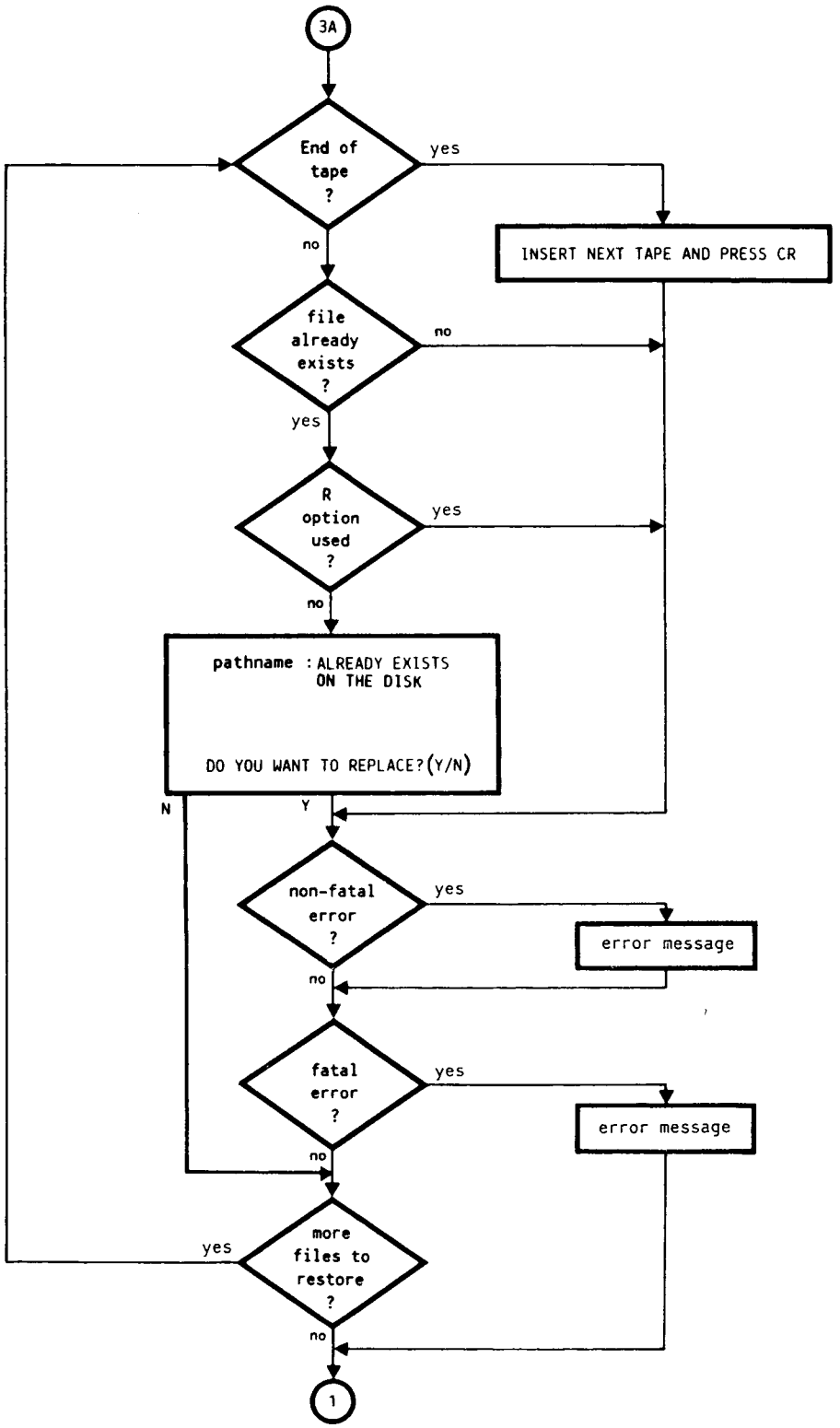


Fig. 3.FILETAR-5 FILETAR Restore Command - Operator Interface

ENTER NAME TO RESTORE:

xx...x The user may enter up to 15 file names to be retrieved from the tape, separated by spaces; the list is completed with **CR.** (See "Optimisation" for information on how to arrange heterogeneous files.)

If the tape contains more than one file bearing a name specified in the restore, only the first one is retrieved.

ENTER PATHNAME ON DISC OR <CR>

xx...x Give full path names for every file to be restored to disc. In a distributed system these can be remote names, defined on another machine in the network.

CR The disc files will be restored with the names that they have on tape.

ENTER OCCURRENCE NUMBER (DEFAULT IS 1):

In naming a file system object that is to be retrieved from tape, the user must identify the version he wants from the several (possible) others of the same name on the tape.

n Restores the "nth" version of the required file (out of the possible several versions on the tape).

CR Entered on its own, restores the first version of the specified file found on the tape.

ENTER DRIVE NUMBER (DEFAULT IS 1):

The user enters the MTU drive number.

n Selects the MTU drive number "n".

CR Selects the MTU drive number 1 by default.

ENTER OPTION(S) (V=VERBOSE/R=REPLACE/U=UNCHANGED)

V The name and size of the file system object being restored is displayed in the format:

RESTORE FROM MTn: xx...x
TYPE: aa...a
SIZE: nn...n TAPE BLOCK(S), ll...l BYTE(S) [,dd...d DIR-ENTRY/IES]
SAVED ON: year month day

R Any object with the same name on the system disc is replaced.

U Enables only one extent to be reserved for the file to be recovered.

CR Neither option is selected.

---

path name: ALREADY EXISTS ON THE DISK DO YOU WANT TO REPLACE? (Y/N)

An attempted restore operation has found that the named file already exists on the disc, but the replace option has not been specified. This is a second chance for the user.

Y Allows the existing object on disc to be overwritten.

N If no more objects were named in the command, returns to the main menu immediately. If another object was named in the same command, its processing begins.

ENCOUNTERED END OF TAPE: UNLOAD (REWIND) THE TAPE

End of tape: wait until the tape has been rewound and unloaded automatically.

INSERT NEXT TAPE AND PRESS <CR>

Wait for the drive light to stop flashing (tape is being rewound), then mount the next tape of the set, set the drive "on-line", and press **CR**.

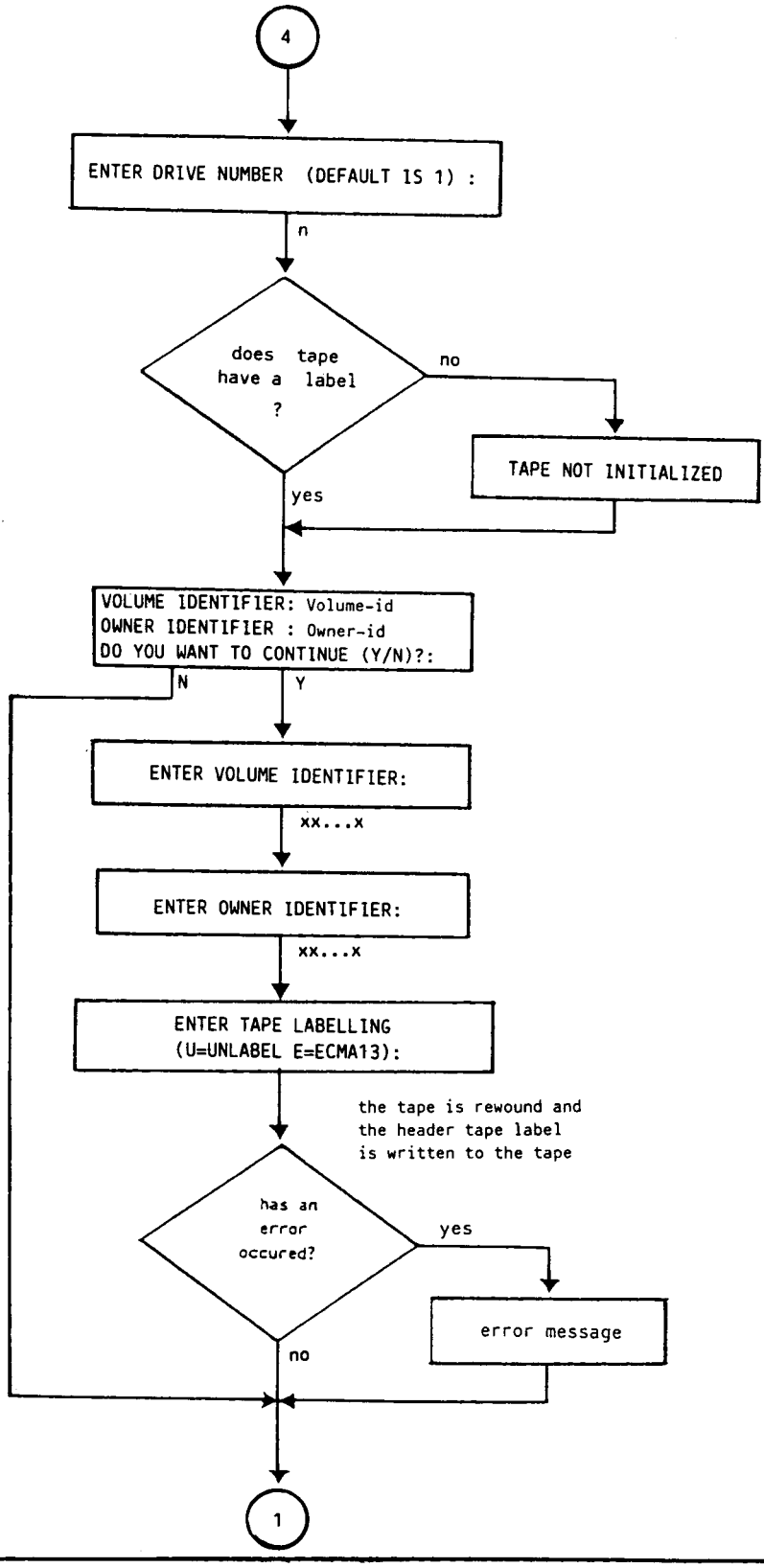


Fig. 3.FILETAR-6 FILETAR Create Command - Operator Interface

ENTER DRIVE NUMBER (DEFAULT IS 1):

The user enters the drive number where the MTU to be initialised is mounted.

n Selects the MTU drive number "n".

CR Selects the MTU drive number 1 by default.

VOLUME IDENTIFIER : volume-id
OWNER IDENTIFIER : owner-id
DO YOU WANT TO CONTINUE (Y/N)?

A header label already exists on the tape, whose details are as shown. This gives the user a chance to double-check on his intentions before overwriting it. If tape label is not standard, the following message is displayed before the confirmation request:

[HEADER TAPE LABEL IS NOT STANDARD]

Y The existing tape header will be overwritten with a new one whose details will be requested later.

N The tape creation function is aborted.

ENTER VOLUME IDENTIFIER:

Begins the prompts which build up the details required for the tape header label.

xx...x Enter the volume name of the data to be saved on tape, (maximum of 6 characters). Longer identifiers are shortened to 6 characters. **CR** completes the entry.

ENTER OWNER IDENTIFIER:

xx...x Identify the tape owner, (maximum of 14 characters). **CR** completes the entry.

ENTER TAPE LABELLING (U=UNLABEL E=ECMA13)

The user is asked how the tape is to be labelled.

U Specifies that whenever a file is written to the tape, only the file identity is to precede it (so called "unlabelled" mode).

E Specifies that whenever a file is written to the tape, a full file header label of ECMA 13 standard is to precede it.

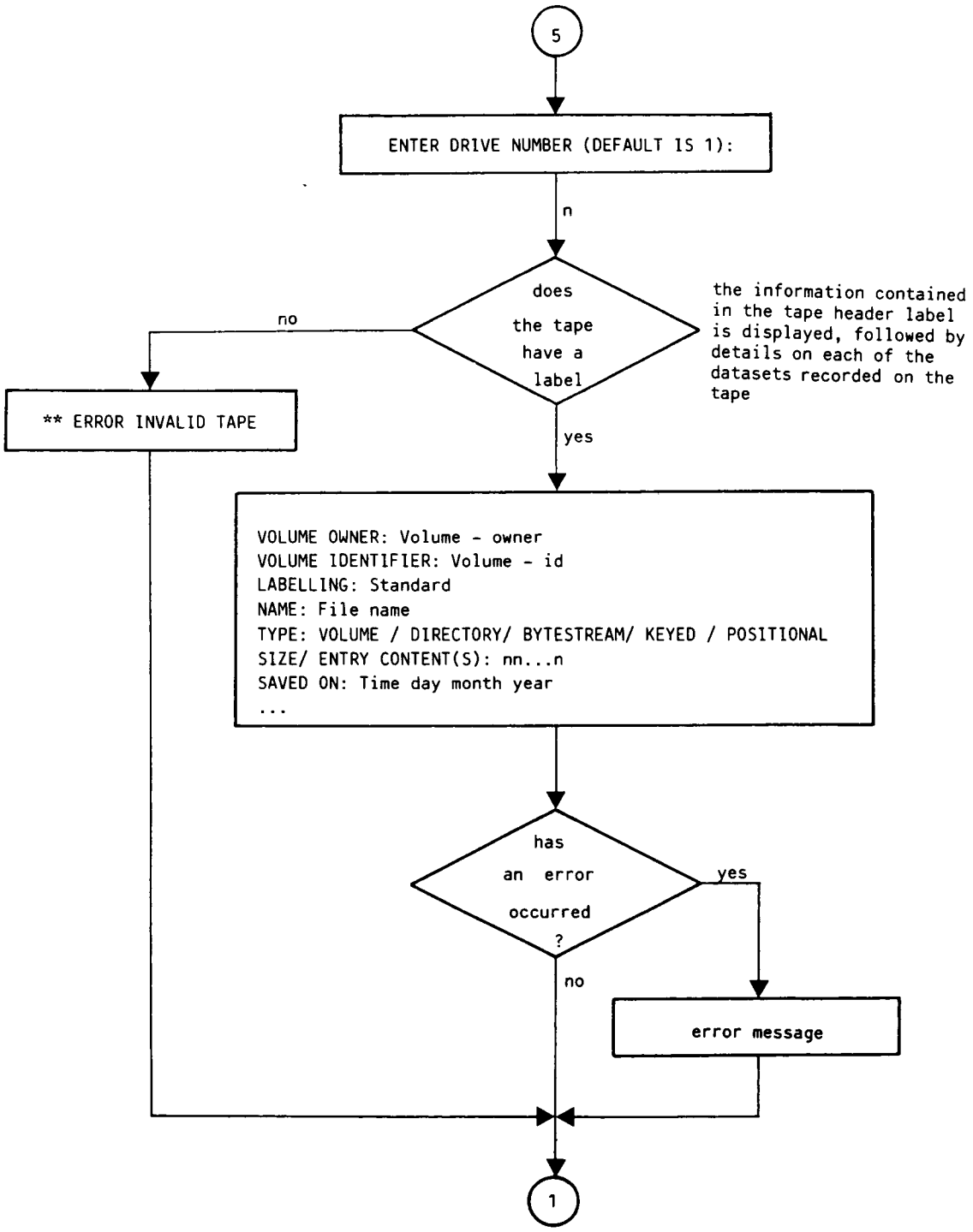


Fig. 3.FILETAR-7 FILETAR Verify Command - Operator Interface

ENTER DRIVE NUMBER (DEFAULT IS 1)

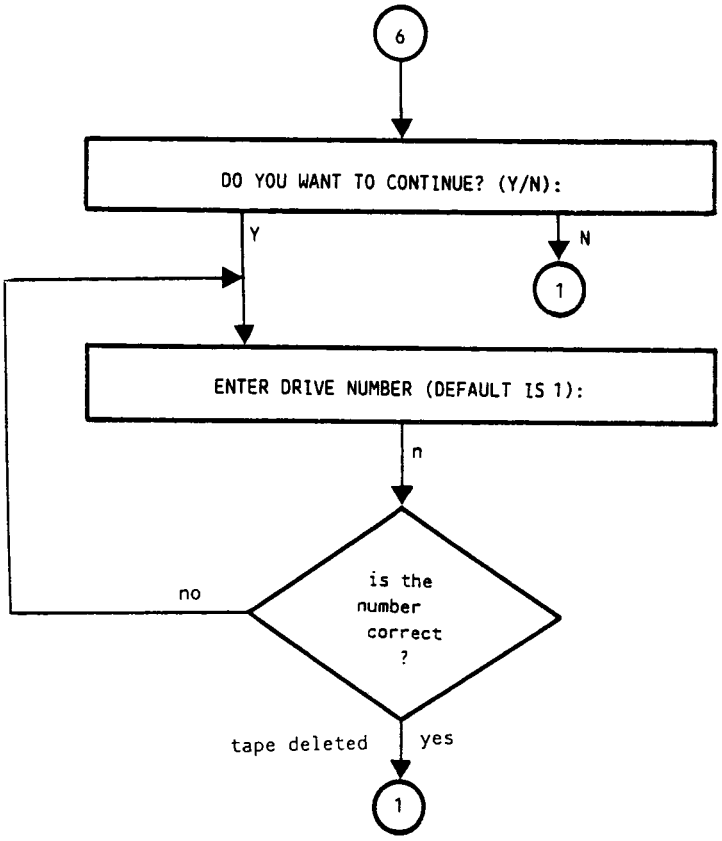
The user enters the MTU drive number.

n Selects the MTU drive number "n".

CR Selects the MTU drive number 1 by default.

```
VOLUME OWNER: owner-id
VOLUME IDENTIFIER: volume-id
LABELLING: standard
NAME : file name
TYPE : VOLUME/DIRECTORY/BYTESTREAM/KEYED/POSITIONAL
SIZE/ENTRY CONTENT(S): nn...n
SAVED ON : time  day  month  year
```

If a tape header label exists, owner, identifier, and label are displayed. Then the name, type, and date of creation of all file system objects on the tape are displayed, as well as the size of the dataset or the number of entries for a directory.



DO YOU WANT TO CONTINUE? (Y/N):

The user is asked to confirm the operation.

N FILETAR asks the user to select another function.

Y The procedure continues.

ENTER DRIVE NUMBER (DEFAULT IS 1)

The user enters the MTU drive number.

n Selects the MTU drive number "n".

CR Selects the MTU drive number 1 by default.

---

path name: ALREADY EXISTS

The file system object specified already exists on disc, and cannot be overwritten because the replace option (**"R**) has not been specified.

CANNOT SAVE: path name

The current save failed because it needed the recursive option **R** , since the path name given identifies a directory containing sub-directories: these can only be dumped if the recursive option is used.

** ERROR: EMPTY TAPE

Restore has aborted because, although correctly labelled, the tape does not contain any data.

** ERROR IN COMPILATION OF TAR LABEL

Save has aborted because a checksum error was found while writing the file header label group.

ERROR IN CONNECT OF file name
ERROR IN OPEN OF file name

No object was found with the name given, probably because it does not correspond to an existing file.

** ERROR IN DIRECTORY CHECK

Save has aborted because a read length error has occurred during the directory read.

** ERROR IN DISK READ - SAVE FUNCTION ABORT <xx...x> IS NOT COMPLETE

Save has aborted because an I/O error occurred during a read operation from the disc. The file system object shown by "xx...x" has not all been saved.

** ERROR IN DISK WRITE

Restore aborted because an I/O error occurred while writing on the disc.

** ERROR IN I/O PROCESS INITIALISATION

An error has occurred during the initialisation of the MTU driver.

** ERROR IN READ OF STANDARD LABEL

Restore aborted because an I/O error occurred while reading the volume header label on tape.

** ERROR IN TAPE OPEN/REWIND

Restore has aborted because a hardware error occurred during a tape rewind operation.

** ERROR IN TAPE UNLOAD

Restore has aborted because an error occurred while the tape was being unloaded.

** ERROR IN TAPE WRITE

An I/O error occurred while trying to write the new tape header label.

** ERROR INVALID TAPE LABEL

Save has aborted because the tape header label is of neither ECMA 13 standard type, nor of unlabelled type.

** ERROR INVALID TAPE LABEL
** LABEL NOT STANDARD, INVALID TAPE

Restore has aborted because the tape header label is not of standard type.

** ERROR IN WRITE OF xx...x LABEL

Save has aborted because an I/O error occurred while writing the tape label specified.

** ERROR ON TAPE ERASE

An I/O error has occurred during tape deletion.

** ERROR ON TAPE OPERATION

An I/O error occurred while trying to read from or write to the tape.

** ERROR ON TAPE READ

The restore, create, or verify function aborted because an I/O error occurred while reading from the tape.

** ERROR ON TAPE REWIND

The save, erase, or verify function aborted because a hardware error occurred during the tape rewind.

** ERROR ON TAPE WRITE

Save has aborted because an I/O error occurred during a write operation on the tape.

** ERROR: VIRGIN TAPE

Restore has aborted because the tape has no tape header label.

** ERROR: WRONG TAPE IDENTIFIER

Restore aborted because the tape header label is different from the identifier entered by the user.

** INDEX OUT FROM PARAM ARRAY : END OF SAVE
** ERROR IN SAVE OF AUXILIARY KEY OR POSITIONAL FILES
** PARAM ARRAY OUT OF BOUNDS : END OF SAVE

Save has aborted because one of the files specified has too long a path name.

** INPUT ERROR: BLOCK MUST BE INCLUDED BETWEEN 512 AND 16384

The block size must be between 512 and 16K bytes.

** INPUT ERROR: BLOCK SIZE MUST BE EVEN

The block size input must be an even number.

** INPUT ERROR: CONFLICT IN PARAMETER NUMBER

The number of file system objects to be read from tape must be the same as that to be stored on disc.

** INPUT ERROR: ILLEGAL PERIPH NUMBER

The peripheral identifier may only be 1 or 2.

** INPUT ERROR: ILLEGAL STRING CONTENT

The input string starts with a blank or CR and is not a default value.

** INPUT ERROR: INVALID BLOCK SIZE

The block size must be a multiple or sub-multiple of 512 bytes.

** INPUT ERROR: NAME(S) TOO LONG

The input string contains one or more path names that are longer than 100 characters.

** INPUT ERROR: NUMBER OF PARAMETERS IS LONGER THAN 15

No more than 15 parameters are allowed in input.

** INPUT ERROR: TWO PARAM NAMES ARE EQUAL

The input string contains two parameters which are the same.

** INSUFFICIENT SPACE AVAILABLE ON THE DISK
** VOLUME CREATION ABORT **

Not enough space remains on the disc for the specified file, so the restore operation terminates.

** INVALID TAPE

Verify aborts because the tape does not contain a tape header label (probably because it has been used for bootstrapping), and is therefore unacceptable.

** PARAM ARRAY OUT OF BOUNDS: END OF RESTORE

During a restore of a positional or keyed file, too long a path name was entered.

** TAPE NAME(S) NOT FOUND: **

1 - param1
. . .
N - paramN

Restore failed because the system is unable to find all tape names. A list of them is displayed showing the sequence numbers of the input parameters, together with the parameters themselves.

** THE CURRENT TAPE SECTION IS INCORRECT

Restore aborts because, in a multi-tape file, the file section is not correct. Check the name and date of the section.

THERE ARE DIFFERENCES AT POSITION nn...n

The user is returned to the main menu, because the tape data at offset position "nn...n" is different from the disc data which should be identical.

** UNKNOWN **

Verify aborts because the tape header label is non-standard in format, and its contents cannot be displayed.

WARNING: ENCOUNTERED A/SOME PARAMETER ERROR(S)

During a save operation, at least one of the specified file system objects has not been successfully saved.

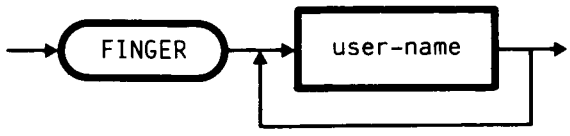
WARNING: I/O OPERATION WILL BE EXECUTED AT HIGH SPEED. BLOCK SIZE WILL BE FORCED AT 15360 BYTES.

The user is informed of the size of the blocks in which data will be stored.

Displays information about one or more users as defined by the reserved command "USERMAN". If the user is logged into the system at the same time that his descriptive information is being displayed with this command, the following information is displayed:

- the login name
- the name and surname of the user who is connected
- the groups he belongs to
- the date and time when he logged in
- the name of the terminal (physical or virtual) at which he logged in.

If the user is not connected to the system, only the date and time are displayed, together with the name of the terminal to which the user was last connected.



---

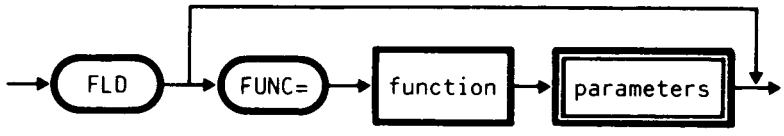
where:

user-name is the name of the user whose information is to be displayed.

Example

```
MCL: FINGER ROOT JIM STEVE

LOGIN NAME: ROOT    IN REAL LIFE: SYSTEM MANAGER
GROUP NAME: SYSTEM
ON SINCE MON OCT 16    10:45:33 1987 ON TTYA

LOGIN NAME: JIM     IN REAL LIFE: JAMES HUSTON
GROUP NAME: TECHNICAL
ON SINCE MON OCT 16    15:15:10 1987 ON TTYC

LOGIN NAME: STEVE   IN REAL LIFE: STEPHEN BRUNT


MCL:_
```

FLD can dump any file, directory, logical or physical volume from a hard disc or 1 Mbyte floppy disc to another 1 Mbyte floppy disc (8" or 5"), and perform the converse restore operation.



where:

function is one of: **S** (logical Save), **R** (logical Restore), **V** (Verify), **C** (Create), **D** (Duplicate floppies), **H** (physical save), or **F** (physical restore).

parameters is the set of parameters that must be specified for each function (see below).

FLD FUNCTIONS

LOGICAL SAVE	Copies a file, directory, logical or physical volume from hard or floppy disc to a backup version on floppy disc.
LOGICAL RESTORE	Restores a file, directory, logical or physical volume on hard or floppy disc, from a backup version stored on floppy disc.
VERIFY	Displays the volume header label of a backup floppy disc, and the details of the file system objects stored on it.
CREATE	Writes a volume header label on cylinder 0 of a floppy disc, to overwrite an existing label or to re-initialise an erased disc for recording data.
DUPLICATE FLOPPY DISCS	Does a physical copy of one floppy disc to another.
PHYSICAL SAVE	Saves the contents of a floppy disc onto hard disc.
PHYSICAL RESTORE	Recovers the contents of a floppy disc from hard disc.

Characteristics

1. A disc created by FLD cannot be interpreted by anything but FLD (it cannot even be mounted using MNT).

2. New discs must be formatted and labelled before being submitted to the create function.

3. FLD can function interactively or non-interactively.

4. If a read error on the source floppy disc occurs while restoring its contents, it is possible to remove the floppy disc, make a copy of it onto another disc, and insert that copy in the same drive. This will enable the restore operation to resume from the point of failure.

5. A floppy can contain more than one file or logical volume, which can be continued on other floppies.

6. The same file name can occur several times on the same floppy disc. Each is distinguishable by its occurrence number.

7. If an object is bigger than 3 blocks, the PACK command, used before the dump, can reduce the disc space occupied by the object. Such packed files may be saved using FLD, then restored to hard or floppy disc in the same format, but must be unpacked before being accessed directly.

8. FLD allows a volume ROOT directory to be saved and restored. This directory can be restored onto a volume of a different size from the original volume.

Warnings

- The backup cannot be used instead of the original; it may only be used to restore the original version.

- FLD cannot be used to restore the contents of a backup created in the previous release with FLDUMP.

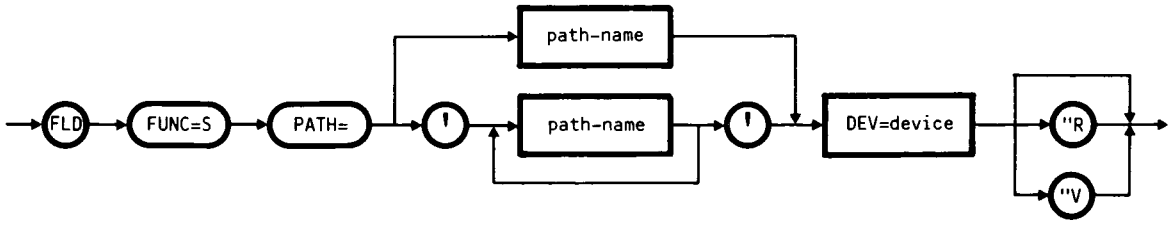
- The KILL command interrupts a save or restore function, but the output of the current dump is lost. Objects dumped previously in the same operation will not be affected.

NON-INTERACTIVE FLD

In non-interactive mode, each function once initiated is carried out to its end without stopping.

Functions and possible parameters follow:

LOGICAL SAVE FUNCTION



where:

path-name specifies the path name of the local source file system object to be saved (not "/IPL", "/.." the memory volume, or a physical device). Multiple path names (maximum of 15) must be separated with a single space and the whole sequence of path names enclosed in single quotes.

device names the destination floppy disc unit (from FL1 to FL4, or MF1 to MF4).

"R requests a recursive save, in the case of a directory: that is, the dump will proceed "vertically" (see Appendix D, "Recursive Organisation"). If not specified, only objects at the first level of the directory will be saved.

"V displays a "trace" of operations performed, as the save proceeds.

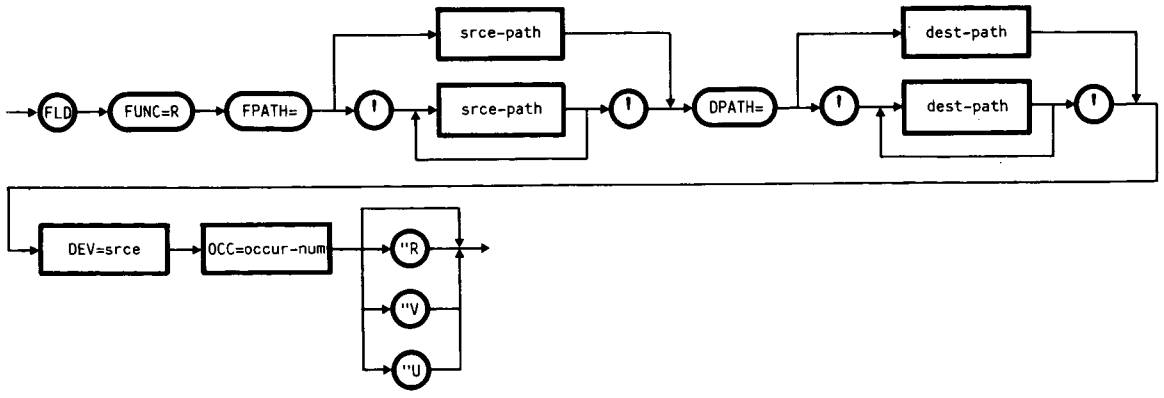
Characteristics

1. Ensure that the system date and time are correct before starting an FLD save, as both are written to the backup disc(s).

2. The dump will be appended to the data already present on the destination floppy: that is, nothing will be overwritten, not even objects with the same name as the current object (unless the destination floppy is re-initialised, effectively deleting everything).

3. A save operation may request more than one floppy disc.

4. The options **"V** and **"R** can be combined in the save operation.



where:

srce-path is the source path name of the file system object to be restored. Multiple path names (maximum of 15) must be separated with a single space and the whole sequence of path names enclosed in single quotes. Remember to specify the path name in exactly the same way as it was entered when it was saved.

dest-path is the destination path name under which the file system object(s) are to be written. Multiple path names (maximum of 15) must be separated with a single space and the whole sequence of path names enclosed in single quotes.

srce names the source floppy disc unit (from FL1 to FL4, or MF1 to MF4).

occur-num is the occurrence number of the object on the source floppy disc.

"R specifies that any existing destination object as specified by **dest-path** is to be overwritten.

"V displays information about the object(s) being restored.

"U allows the object to be restored maintaining the same characteristics as it had when it was saved (the length of the first extent remains unchanged).

Characteristics

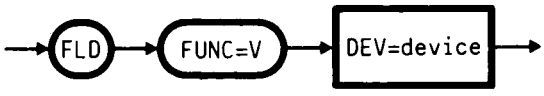
1. Usually, the restored objects are assigned the same access rights as defined in the default access list. The exceptions to this follow:

   - If the option **"R** is specified when restoring a directory, its access rights are not changed from those of the existing directory.

   - When a volume is being restored, the access rights and the owner of the object in it are unchanged.

2. The allocation unit of a restored file is not changed, even if the option **"U** is specified.

VERIFY FUNCTION



where:

device names the floppy disc unit (from FL1 to FL4, or MF1 to MF4).

Characteristics:

1. Every file on the floppy is shown separately.

2. Each occurrence of the same file name is displayed separately, since each is a separate file.

CREATE FUNCTION

where:

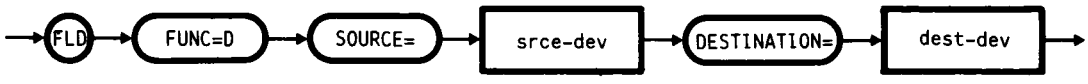
device specifies the floppy disc unit (from FL1 to FL4, or MF1 to MF4) where the disc to be labelled is mounted.

floppy-name is the name of the floppy disc, as a string of 14 characters maximum.

owner-name identifies the owner of the floppy disc (maximum of 8 characters).

Note: Remember that all the data already present on the disc will be logically deleted by this operation.

FLOPPY DISC DUPLICATION FUNCTION



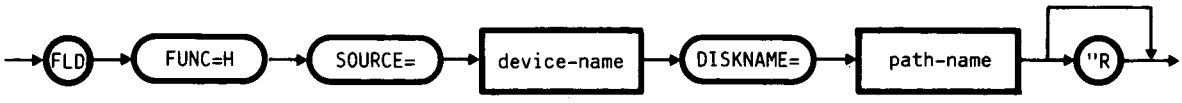
where:

srce-dev is the name of the source floppy or mini-floppy disc unit (from FL1 to FL4, or MF1 to MF4).

dest-dev is the name of the destination floppy or mini-floppy disc unit (from FL1 to FL4, or MF1 to MF4).

Note: The two floppy or mini-floppy discs must have the same format.

PHYSICAL SAVE FUNCTION



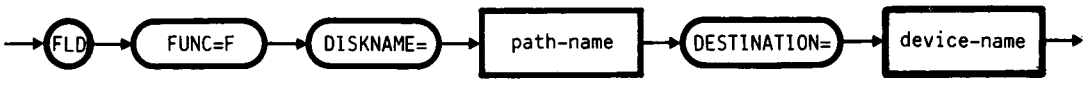
where:

device-name is the name of the floppy or mini-floppy disc whose contents must be saved on hard disc (from FL1 to FL4, or MF1 to MF4).

path-name is the name or path name of the file in which the contents of the floppy or mini-floppy disc are to be saved.

"R is the option with which the file can be overwritten, if it is on the hard disc already.

PHYSICAL RESTORE FUNCTION



where:

path-name is the name or path name of the hard disc file whose contents are to be restored onto floppy or mini-floppy disc (from FL1 to FL4, or MF1 to MF4).

device-name is the name of the destination floppy or mini-floppy disc unit (from FL1 to FL4, or MF1 to MF4).

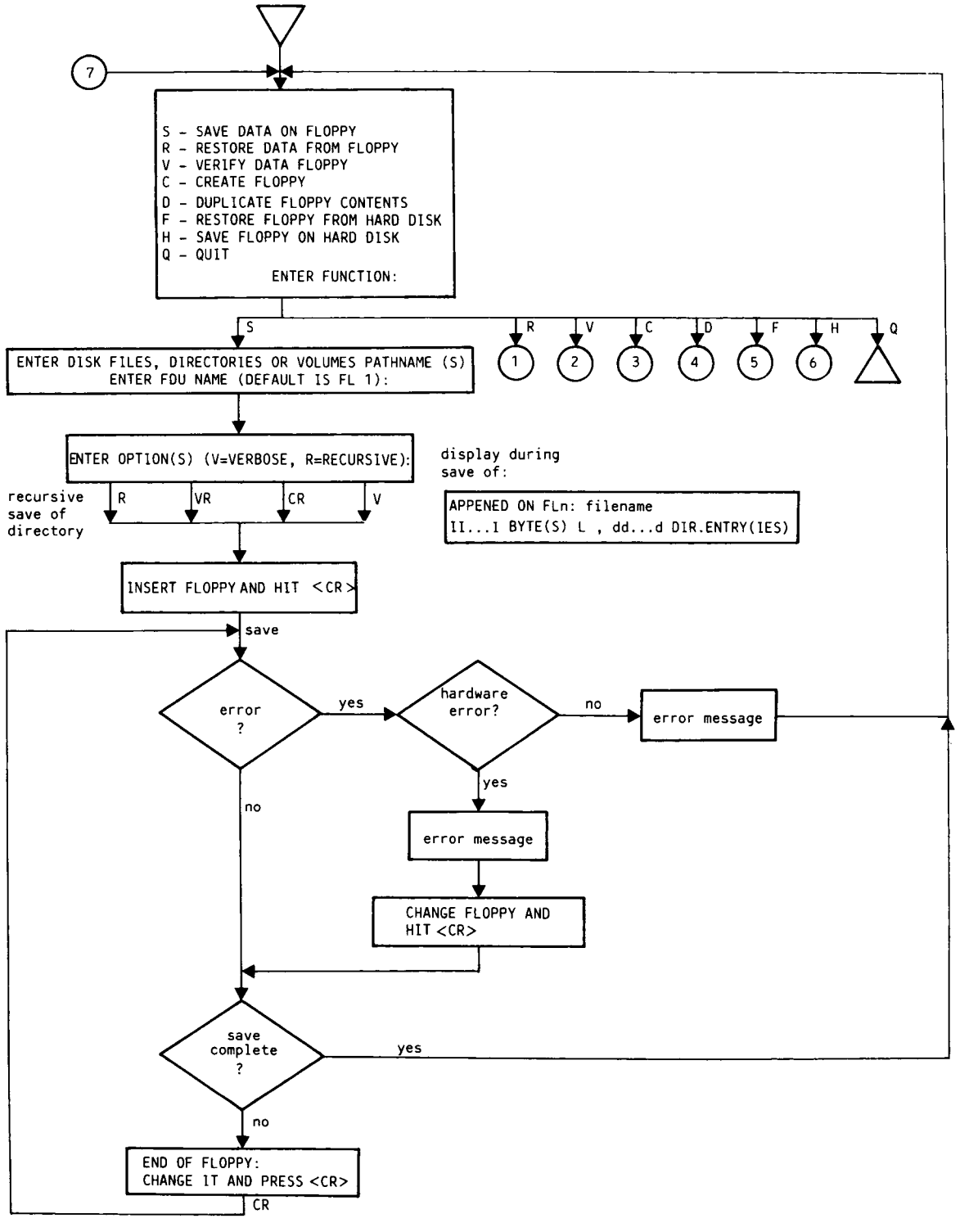


Fig. 3.FLD-1 FLD Command - Interactive Operator Interface (cont.)

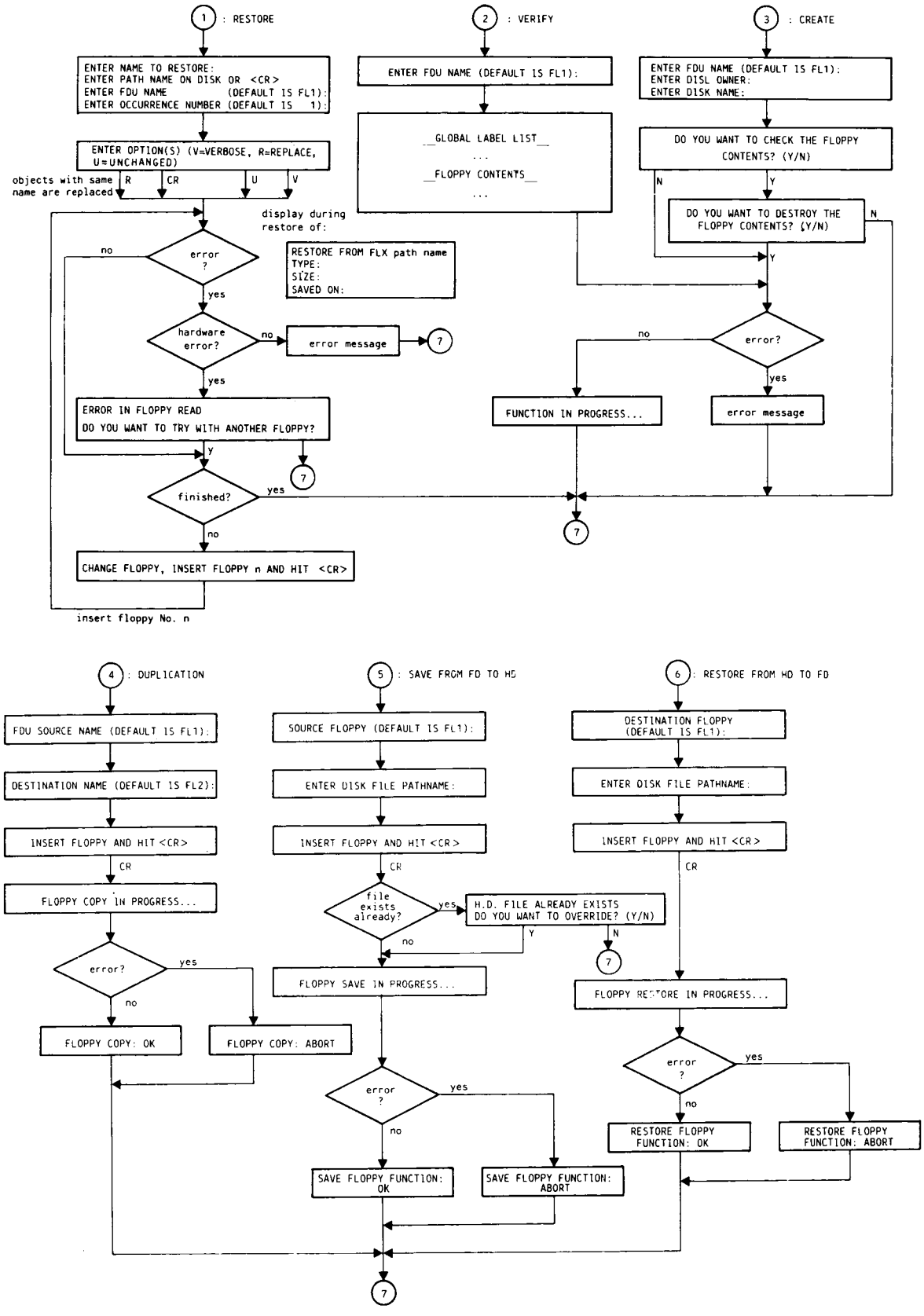


Fig. 3.FLD-2 FLD Command - Interactive Operator Interface

INITIAL DISPLAY

S - SAVE DATA ON FLOPPY

R - RESTORE DATA FROM FLOPPY

V - VERIFY DATA FLOPPY

C - CREATE FLOPPY

D - DUPLICATE FLOPPY CONTENTS

F - RESTORE FLOPPY FROM HARD DISK

H - SAVE FLOPPY ON HARD DISK

Q - QUIT

ENTER FUNCTION:

LOGICAL SAVE FUNCTION

ENTER DISK FILES, DIRECTORIES OR VOLUMES PATHNAME(S):

Enter the path name of the file, directory or volume, to be dumped to floppy disc. If required, specify more than one path name (maximum of 15), each separated by one space.

CR Completes the entry.

Note

- The current directory, if required, can be indicated as ".".
- If a directory name is entered, only files at its first level will be saved unless the recursive option is used (see below).
- Ensure that the system date and time are correct before starting a save, since both are written to the floppy disc(s).
- Do not try to save a physical device, the memory volume, "/.", or "/.."

ENTER FDU NAME (DEFAULT IS FL1):

Enter the identifier of the destination drive to be used: FLn or MFn, where disc type is FL or MF and "n" is the drive number; the default is "FL1".

CR Completes the entry, or if entered on its own, selects the default drive FL1.

ENTER OPTION(S) (V=VERBOSE, R=RECURSIVE):

V Displays the details of the save in progress.

R Requests recursive operation, to save all the levels of a directory vertically (see Appendix D, "Recursive Organisation").

VR Combines the above two options.

CR Selects neither option if entered on its own.

INSERT FLOPPY AND HIT <CR>

Insert the first destination floppy and press **CR.**

CHANGE FLOPPY AND HIT <CR>

Insert another source floppy and press **CR.**

END OF FLOPPY: CHANGE IT AND PRESS <CR>

Insert the next destination floppy and press **CR.**

SAVE FUNCTION: OK

• The save function has completed successfully.

ENTER NAME TO RESTORE:

Enter the path names of the source files to be restored to hard or floppy disc (maximum of 15).

CR Completes the entry.

ENTER PATHNAME ON DISC OR <CR>:

Enter the destination path names of the objects, if these are different from the source path names. Do not restore remote objects, or path names "/IPL" and "/.".

CR Completes the entry, or if entered on its own, restores the objects with the same path names as they have on floppy disc; these are the names they had when they were saved from the current source disc originally.

ENTER FDU NAME (DEFAULT IS FL1):

Enter the name of the source drive: MFn or FLn, where MF or FL is the drive type, and "n" is the drive number.

CR Completes the entry, or if entered on its own, selects the default drive FL1.

ENTER OCCURRENCE NUMBER (DEFAULT IS 1):

The occurrence number on the floppy of the file system object to be restored.

CR If entered on its own, selects the first occurrence by default, restoring from the first occurence of the object on the source floppy.

ENTER OPTION(S) (V=VERBOSE, R=REPLACE, U=UNCHANGED)

V	Displays the details of the restore in progress.
R	Allows the overwriting of the specified files, directories, or volumes already present on the destination disc; however, the source and destination files must be of the same type (for instance, byte-stream). If not, the overwrite is prevented.
U	Allows a file to be restored maintaining the same characteristics as it had when it was saved.
CR	Selects neither option if entered on its own.

CHANGE FLOPPY, INSERT FLOPPY n AND HIT <CR>

All the data has been restored from the current source disc, and the system requires the next disc of the set to be mounted.

CR	Resumes the restore operation.

RESTORE FUNCTION: OK

The data restore function has completed successfully.

ENTER FDU NAME (DEFAULT IS FL1):

Specify the disc drive containing the floppy whose contents are to be checked: FLn or MFn, where FL or MF is the disc type, and "n" is a number.

CR Completes the entry, or if entered on its own, selects the default drive FL1.

-- GLOBAL LABEL LIST --

FLOPPY OWNER : owner-id
FLOPPY IDENTIFIER:floppy-id

-- FLOPPY CONTENTS --

NAME: path name
TYPE: VOLUME/DIRECTORY/BYTE-STREAM/KEYED/POSITIONAL
SIZE/ENTRY CONTENTS(S): nn...n
SECTION: secnum
SAVED ON: time and date

The above details correspond to the information held in the floppy header label. They are indicated for each object contained in the floppy: the path name of the saved file is shown, together with its type below; "nn...n" represents the length of the data saved in bytes, or the number of elements in the case of a saved directory; "secnum" is the sequence number of the dataset on the floppy.

DO YOU WANT TO CONTINUE?

N If the display requires more than one screen page, this allows the user to dismiss the remainder.

Y To continue with the display.

ENTER FDU NAME (DEFAULT IS FL1):

Specify the disc drive where the floppy is mounted: FLn or MFn, where FL or MF is the disc type, and "n" is a number.

CR Completes the entry, or if entered on its own, selects the default drive FL1.

ENTER DISC OWNER (8 CHARS):

Specify the name of the owner of the floppy disc (maximum of 8 characters), completing the entry with **CR**.

ENTER DISC NAME (14 CHARS):

Enter the floppy disc name (maximum of 14 characters), completing the entry with **CR**.

DO YOU WANT TO CHECK THE FLOPPY CONTENTS? (Y/N)

Y Displays the contents of the destination disc.

N Continues with the actual label creation stage.

DO YOU WANT TO DESTROY THE FLOPPY CONTENTS? (Y/N)

Y Continues with the create function.

N Returns to the main menu.

FDU SOURCE NAME (DEFAULT IS FL1):

Specify the disc drive where the source floppy is mounted: FLn or MFn, where FL or MF is the disc type, and "n" is a number.

CR Completes the entry, or if entered on its own, selects the default drive FL1.

DESTINATION NAME (DEFAULT IS FL2):

Specify the disc drive where the destination floppy is mounted: FLn or MFn, where FL or MF is the disc type, and "n" is the unit number.

CR Completes the entry, or if entered on its own selects the default drive FL2.

INSERT FLOPPY AND HIT <CR>

The floppy or mini-floppy must be inserted in the selected drive and **CR** pressed.

FLOPPY COPY IN PROGRESS ...

The floppy or mini-floppy disc is being copied.

FLOPPY COPY: OK

The copying function has completed successfully.

FLOPPY COPY: ABORT

I/O errors have occurred when copying the floppy or mini floppy.

SOURCE FLOPPY (DEFAULT IS FL1):

Specify the disc drive where the source floppy is mounted: FLn or MFn, where FL or MF is the disc type, and "n" is the unit number.

CR Completes the entry, or if entered on its own, selects the default drive FL1.

ENTER DISK FILE PATH NAME:

Indicates the name or path name of the hard disc file, in which the contents of the floppy disc are to be saved.

CR Completes the entry.

INSERT FLOPPY AND HIT <CR>

Insert the floppy in the selected drive and press **CR.**

HD FILE ALREADY EXIST, DO YOU WANT TO OVERRIDE? (Y/N)

The file in which the contents of the floppy or mini-floppy are to be saved already exists on the hard disc. Possible action is:

Y to overwrite the contents of the existing hard disc file with those of the floppy

N to interrupt the save operation and return to the main menu.

FLOPPY SAVE IN PROGRESS ...

The save operation of the floppy onto hard disc is in progress.

SAVE FLOPPY FUNCTION: OK

The save function has completed successfully.

SAVE FLOPPY FUNCTION: ABORT

I/O errors have occurred when saving the floppy on hard disc.

PHYSICAL RESTORE FUNCTION

DESTINATION FLOPPY (DEFAULT IS FL1):

Specify the disc drive where the destination floppy is mounted: FLn or MFn, where FL or MF is the disc type, and "n" is the unit number.

CR Completes the entry, or if entered on its own, selects the default drive FL1.

ENTER DISK FILE PATH NAME:

Prompts for the name or path name of the file on hard disc, whose contents are to be restored onto floppy.

CR Completes the entry.

INSERT FLOPPY AND HIT <CR>

Insert the floppy disc in the selected drive and press **CR.**

FLOPPY RESTORE IN PROGRESS ...

The restore operation onto floppy disc is in progress.

RESTORE FLOPPY FUNCTION: OK

The restore operation onto floppy disc has completed successfully.

RESTORE FLOPPY FUNCTION: ABORT

I/O errors have occurred when restoring the file onto floppy disc.

---

name : ALREADY EXISTS ON THE DISK

The file system object to be restored exists already on the disc, but the replace option has not been specified.

BAD DESTINATION FORMAT

Destination floppy not physically compatible with the peripheral hardware characteristics.

BAD SOURCE FORMAT

Source floppy not physically compatible with the peripheral hardware characteristics.

CANNOT SAVE DIRECTORY: path name

The directory specified contains files at a sub-level (that is, it is recursively organised), but the recursive option (**"R**) has not been specified; FLD only saves the files on the first level of the directory.

ERROR IN COMPILATION OF FLD-LABEL

The two fields that make up the label of the floppy disc have not been compiled correctly.

ERROR IN CONNECT file-name

The object that is to be saved or restored is not on the disc.

ERROR IN CONNECT OF /DEV/Fln

This message appears if an attempt is made to use a floppy mounted with the command MNT.

ERROR IN DIRECTORY READ

Read error in a file of the directory to be dumped.

ERROR IN DISK OWNER LENGTH

The owner name entered is too long (see Appendix B).

ERROR IN DISKNAME LENGTH

The disc name entered is too long (see Appendix B).

ERROR IN FLOPPY READ

A hardware error has occurred while reading the source floppy, which causes the following message to appear:

DO YOU WANT TO TRY WITH ANOTHER FLOPPY?

Having produced this message, the system waits for a reply. The user can try to overcome this error by dismounting the suspect floppy, then trying to copy it on another machine using low-level programs (see "Functional Checks Manual").

N Aborts the restore and returns to the main menu.

Y Resumes the function, when the user has recreated and mounted a new copy of the source disc.

ERROR IN FLOPPY WRITE
FLOPPY LENGTH HAS BEEN REDUCED FOR AN I/O ERROR

A hardware error has occurred during a save operation. The user is prompted to change the destination floppy by:

CHANGE THE FLOPPY AND HIT <CR>

Saving continues when the floppy is changed and **CR** has been pressed.

ERROR IN FLOPPY WRITE
WRONG FLOPPY

A hardware error occurred whilst writing the global label. To continue with the save, the user is requested to:

CHANGE THE FLOPPY AND PRESS <CR>

ERROR IN RESTORE OF AUXILIARY KEY OR POS FILES
PARAM ARRAY OUT OF BOUNDS: END OF RESTORE

A path name longer than allowed (see Appendix B) has been obtained during restore of a keyed or positional file.

ERROR IN SAVE OF AUXILIARY KEY OR POS FILES
PARAM ARRAY OUT OF BOUNDS : END OF SAVE

During dumping of a keyed or positional file, a path name longer than allowed, (see Appendix B), has been obtained for the auxiliary files associated with the primary file.

ERROR IN SEQUENCE NUMBER. CHANGE FLOPPY AND HIT <CR>

The source floppy just mounted is not the one which should follow the previous source floppy. (Floppies must be mounted in the same order as they were in the save which created them).

ERROR IN WRITE OF DESTINATION AT POSITION 0

Hardware error on the destination floppy at the first access to track 0.

ERROR IN WRITE OF FLOPPY LABEL
FLOPPY LENGTH HAS BEEN REDUCED FOR AN I/O ERROR

A hardware error has occurred while writing a label other than the first. To continue the save, the user is requested to:

CHANGE THE FLOPPY AND PRESS <CR>

ERROR IN WRITE OF FLOPPY LABEL
WRONG FLOPPY

A hardware error has occurred whilst writing the first label of a file. To continue the save, the user is requested to:

CHANGE THE FLOPPY AND PRESS <CR>

FLOPPY DISK IS EMPTY - RESTORE FUNCTION ABORT

The source disc is empty, so the attempted restore ends.

FLOPPY NAME(S) NOT FOUND - RESTORE FUNCTION ABORT
1 - file-name-a
2 - file-name-b

The object name(s) shown here and specified in the attempted restore are not on the source floppy.

***FLOPPY NOT INITIALISED WITH CREATE FUNCTION! SAVE FUNCTION ABORT

The floppy disc specified is not formatted. If it is the intended disc, use the FLD CREATE function to label and format it, otherwise mount the correct disc. In either case the save must be restarted.

INCOMPATIBLE FORMAT BETWEEN SOURCE AND DESTINATION

The physical format of the floppy to be copied is not the same of the destination one.

INVALID OPERATION : FILE TYPES ARE INCOMPATIBLE

The restore operation requested with the replace option is not allowed, as the source file type is different from the destination file type.

A built-in check ensures that overwriting, even when the replace option is specified, only takes place when both source and destination file are of the same type, from:

- keyed files
- positional files
- byte-stream files
- directories
- volumes.

PATH NAME TOO LONG: END OF SAVE

The name(s) specified is too long (see Appendix B).

SOURCE/DESTINATION IS NOT READY, CHECK IT AND PRESS <CR>

The source or destination floppy disc has not been inserted correctly in the drive.

TYPE: UNKNOWN

An incorrect file type has been found during reading of the file label.

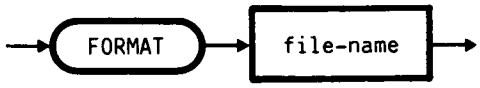
UNRECOVERABLE I/O ERROR ON xxx AT POSITION n

Hardware error on the specified device (FLx or MFx) at position n.

Enables a keyed file or a positional file with record deletion permitted to be converted from 5.2 format to 6.0 format, so that it becomes compatible with the file system of release 6.0.

This command is issued on a machine at release 6.0.

There is a command TAMROF, to perform the converse operation of that of FORMAT.



where:

file-name is the name or path name of the file.

Warning

Before performing the FORMAT utility on the file, the following operations should be carried out:

1. Check file consistency using the appropriate utility, as if FORMAT is performed on an unsuitable file, inconsistencies will occur that cause loss of data.

2. Check that there is a back-up copy of the file, because a system crash during execution of FORMAT could damage the original file.

Characteristics

The following access rights are required to perform the command:

- append or execution and write on the father directory

- write and read on the file.

Error Messages

DEVICE NOT READY

The hard disc or floppy disc drive is off or disconnected, or the floppy disc is not inserted correctly.

HARDWARE ERROR

A hardware fault has occurred.

INVALID ID

The identifier of the specified file is not valid.

INVALID OPERATION

The operation requested is not valid.

INVALID PARAMETERS

The command parameters are not correct.

INVALID PATHNAME

The specified path name is not valid.

KEY ALREADY EXISTS

The file primary index contains duplicate keys.

NAME NOT FOUND

The file name given in the command does not exist.

OUT OF BOUNDS

The position given for the byte or record to read is outside the range of the file.

OUT OF DISK SPACE

There is not enough disc space for the operation requested to be carried out.

SECURITY VIOLATION

The access rights on the file or father directory are incompatible with the kind of operation requested in the command.

SYSTEM ERROR

A fatal error has occurred during execution of the command.

TIMEOUT

The timeout declared when opening the file has expired, and the operation requested has not been executed.

Displays the file HELP.DATA which contains a brief description of the commands in alphabetical order.

A requirement for the command to be executed is that HELP.DATA is present in the directory /IPL/DPC/CMD.

The operator interface for the command HELP is the same as that described for the command MORE.

This command has no parameters.

Characteristics

This command is an MCL procedure using the command MORE on the file HELP.DATA. The user may modify the contents of the file HELP.DATA depending on local requirements.

Aborts a user-activated job (program, command, or procedure) in execution. Shell will resume control over the terminal as if the job had correctly terminated.

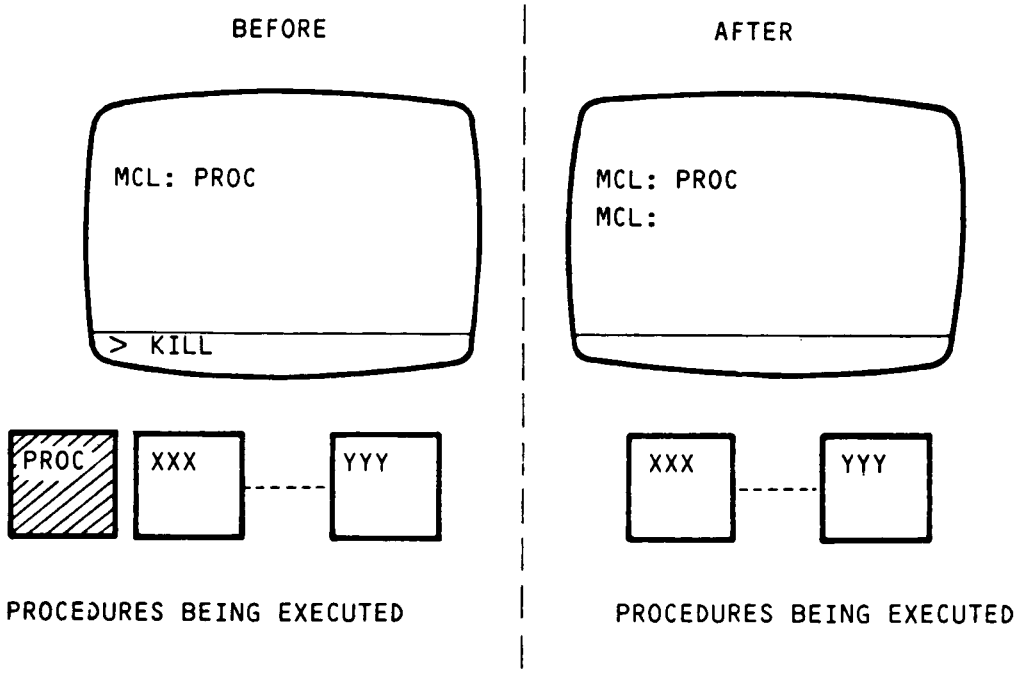
This command must be entered on the system line.

This command has no parameters.

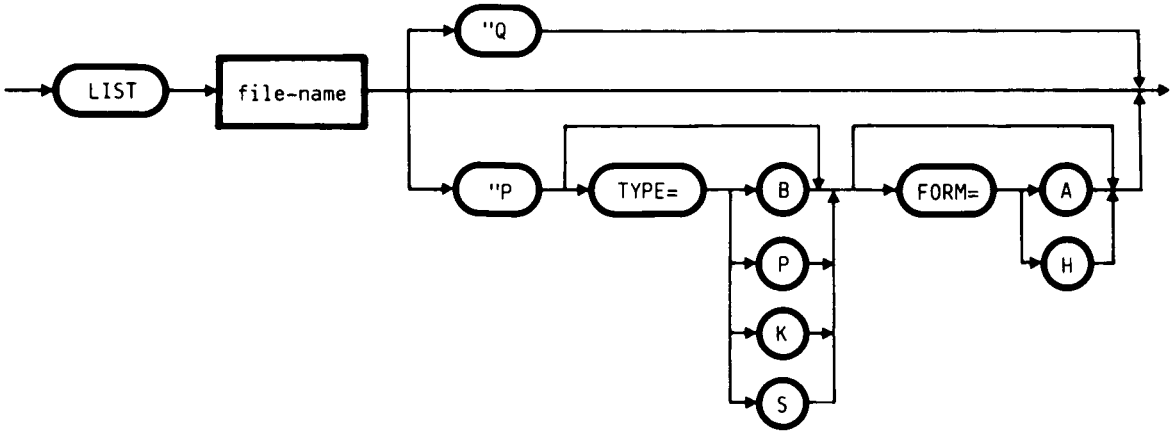
Note

See also the commands: SUSPEND, RESUME.

Example



Displays the contents of a specified file, one page (23 lines) at a time. On the last line of each screen page, the user is prompted whether to continue the display or not by means of a suitable message.



where:

file-name is the name or path name of the file to be listed.

"Q is the interactive option (see operator interface below).

"P is the total display option.

TYPE is the file access type, which may be one of: **B**, **P**, **K**, or **S** indicating a byte-stream, positional, keyed file, or sequential file respectively.

FORM is the display format, which may be one of two possible character codes: **A** for an ASCII display, and **H** for a hexadecimal display.

Characteristics

1. If none of the command options are used, the file is displayed in ASCII code and sequential mode.

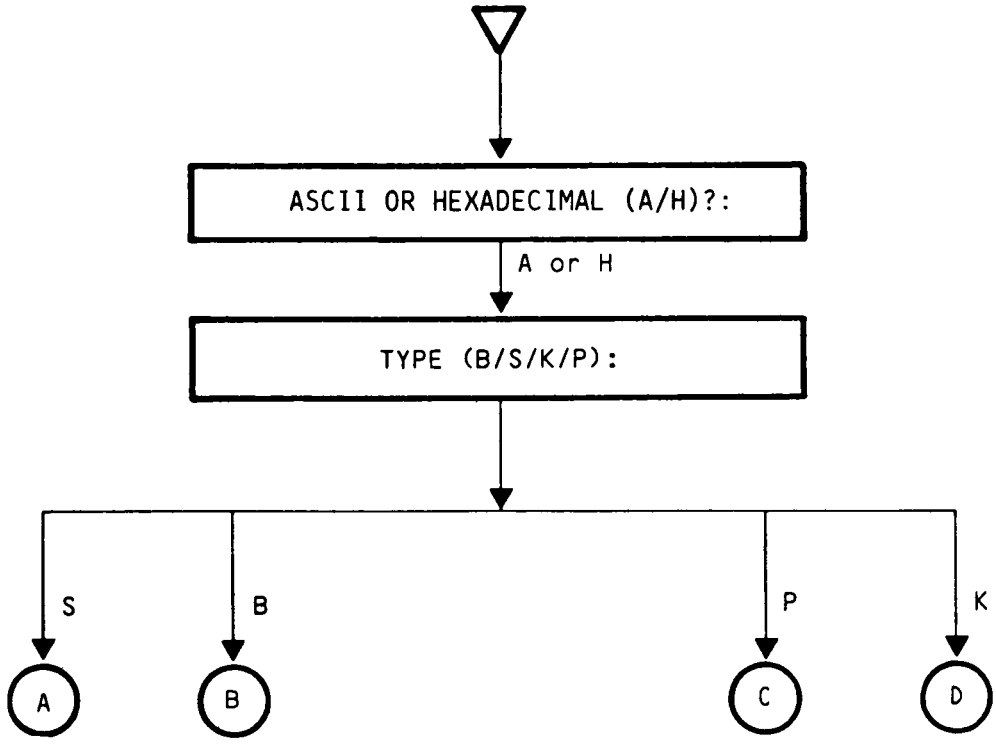
2. The option **"Q** enters interactive mode, producing a series of prompts which vary according to the type of file. This option allows the display of sequential, byte-stream, positional, or keyed files, in ASCII codes or hexadecimal display.

3. The total display option **"P** scrolls through the entire file immediately without stopping. The user may also use the parameters to define a display in different format (access mode) and character code. Otherwise, the default options are display in ASCII codes and sequential mode. The output may be redirected to a printer.

4. Access rights to execute the father directory and to read the file to display are required to execute this command.

Example

```
LIST FRANK/PRINT1 "P TYPE=B FORM=A > /DEV/SYSPRT1
```

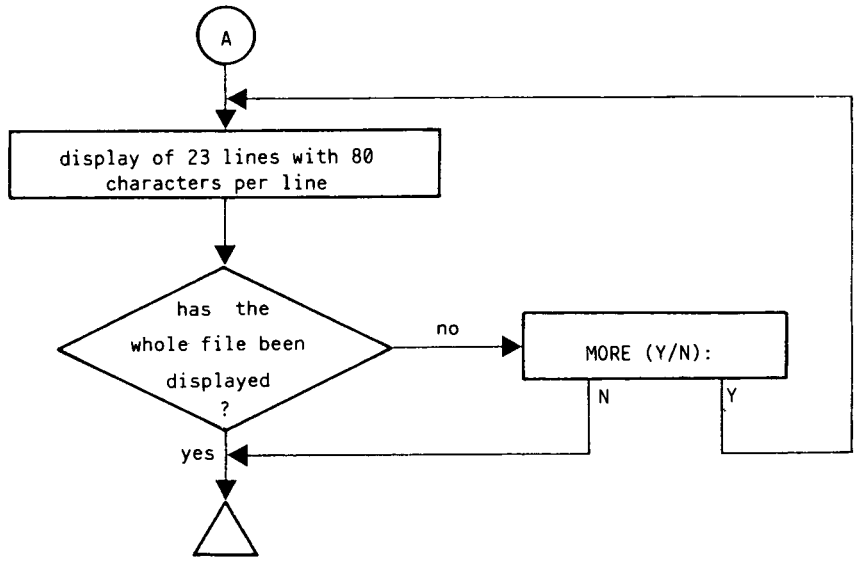


ASCII OR HEXA (A/H)?:

A	Displays the file in ASCII code.
H	Displays the file in hexadecimal code.

TYPE (B/S/K/P):

B	Displays the file in byte-stream mode.
S	Displays the file in sequential mode.
P	Displays the file in positional mode.
K	Displays the file in keyed mode.



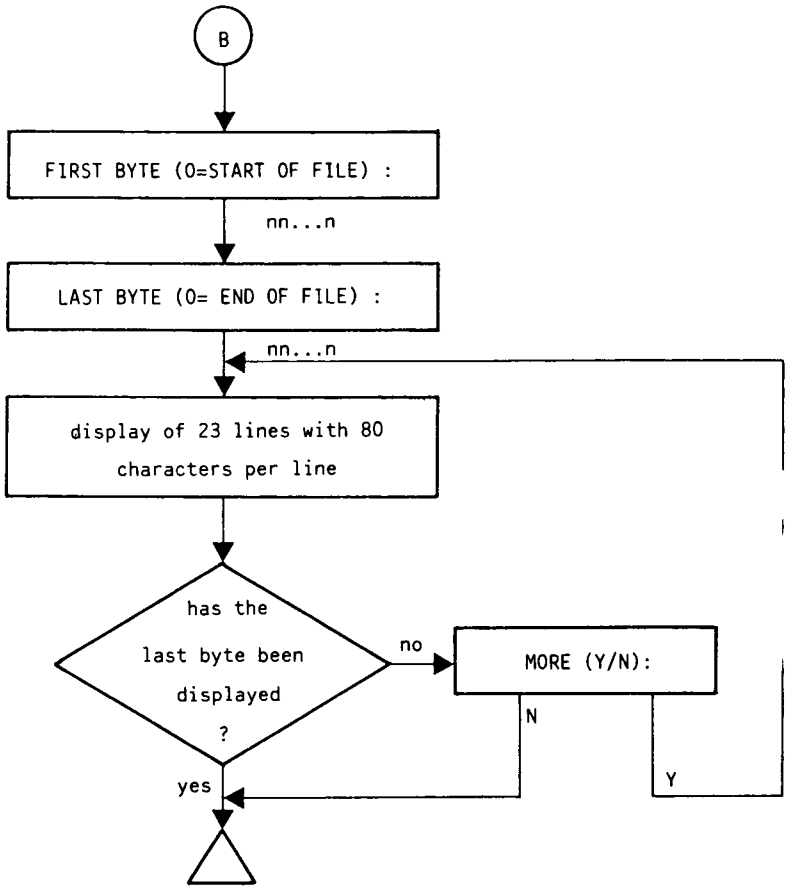
MORE (Y/N):

This message appears on the last line of the screen page. The user must specify whether he wishes the display to continue.

N The display is terminated and the system returns to the Shell environment.

Y The display continues.

Displaying a File in Byte-Stream Mode



FIRST BYTE (0 = START OF FILE):

nn...n The user must enter an integer to indicate the position of the first byte.

If 0 or an invalid value is specified the file is displayed from the beginning.

LAST BYTE (0 = END OF FILE):

nn...n The user must enter an integer to indicate the position of the last byte.

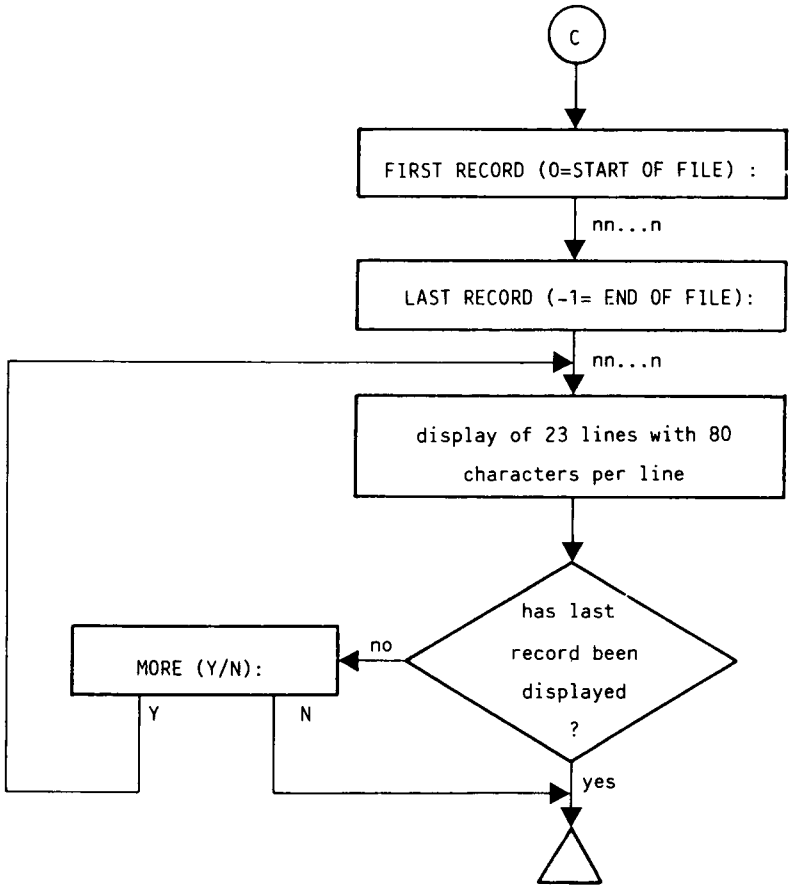
If 0 or an invalid value is specified, the file is displayed to its end.

MORE (Y/N):

This message appears on the last line of the screen page. The user must specify whether he wishes the display to continue.

N The display is terminated and the system returns to the Shell environment.

Y The display continues.


C
FIRST RECORD (0=START OF FILE) :
nn...n
LAST RECORD (-1= END OF FILE):
nn...n
display of 23 lines with 80 characters per line
has last record been displayed ?
no
MORE (Y/N):
Y
N
yes

FIRST RECORD (0 = START OF FILE):

nn...n The user must enter the position of the first record from which display is to start.

If 0 or an invalid value is specified the file is displayed from the first record.

LAST RECORD (-1 = END OF FILE):

nn...n The user must enter the position of the last record to be displayed.

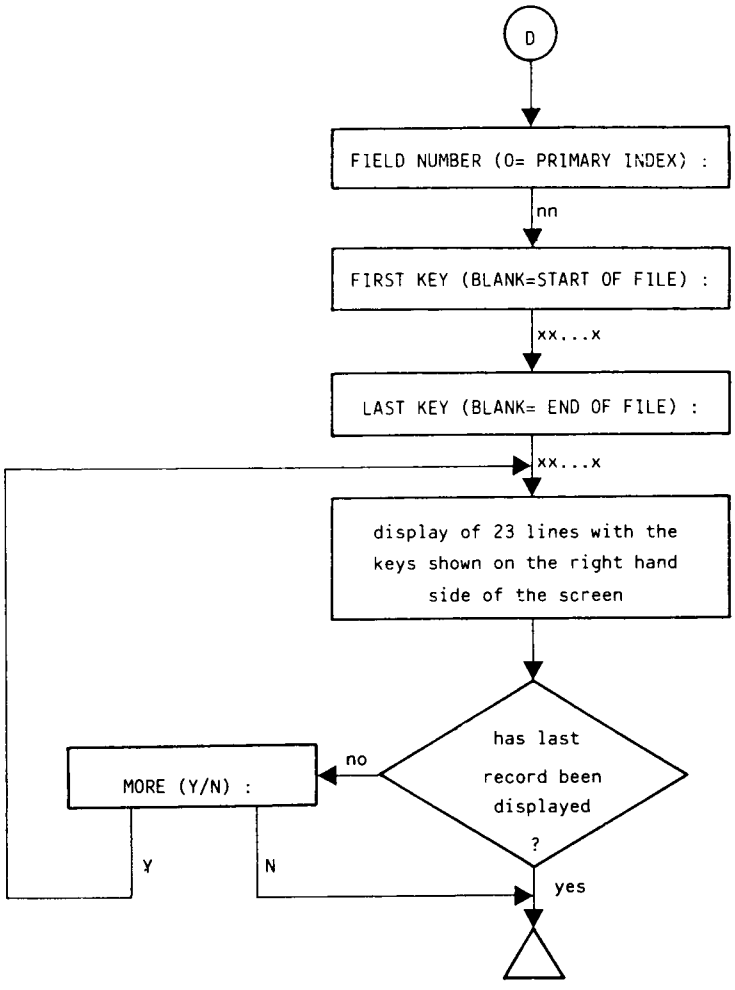
If -1 or an invalid value is specified, the file is displayed to its end.

MORE (Y/N):

This message appears on the last line of the screen page. The user must specify whether he wishes the display to continue.

N The display is terminated and the system returns to the Shell environment.

Y The display continues.


D
FIELD NUMBER (0= PRIMARY INDEX) :
nn
FIRST KEY (BLANK=START OF FILE) :
xx...x
LAST KEY (BLANK= END OF FILE) :
xx...x
display of 23 lines with the keys shown on the right hand side of the screen
has last record been displayed ?
no
MORE (Y/N) :
Y
N
yes

FIELD NUMBER (0=PRIMARY INDEX):

nn The user must enter the index number whose keys are to be used to display the record of the file. The index number can be obtained through the PRY command.

If the user specifies 0, the primary index keys of the file are displayed.

FIRST KEY (BLANK = START OF FILE):

xx...x The user must enter the value of the first key from which the file display is to start.

If the user specifies a blank character or an non-existent key, the file is displayed from the first key.

LAST KEY (BLANK = END OF FILE):

xx...x The user must enter the value of the last key in the file to display.

If the user specifies a blank character or a non-existent key, the file is displayed to the end.

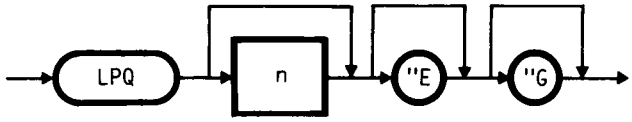
MORE (Y/N):

This message appears on the last line of the screen page. The user must specify whether he wishes the display to continue.

N The display is terminated and the system returns to the Shell environment.

Y The display continues.

Displays the status of a printer and the spool class associated with it.



where:

n specifies a spool class from 1 to 16, for which the following information is displayed: the name and identifier of the unspooler, the status, priority, owner, number of copies, and title (only the first 18 characters) of each job.

"E only lists the unspooler identifier, the total number of characters of the class (including the copies), and the whole title (40 characters at most) of each job in the queue.

"G is an option which, in a distributed system, allows the display of the state of the global printers and their corresponding spool classes.

Characteristics

1. The parameters make it possible to display different amounts of detail for the print jobs in each class queue.

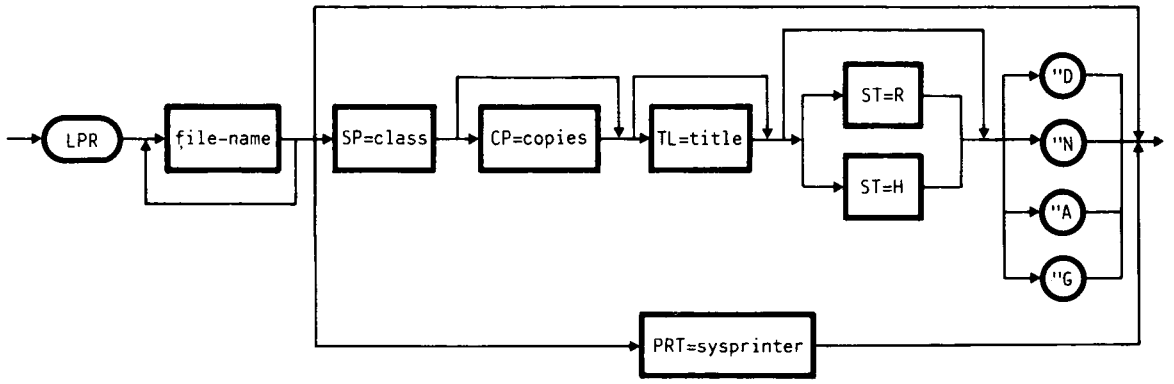
2. When no parameters are supplied, LPQ displays the status of the spool classes associated with the local printers.

3. The information displayed is explained in the SPOOL command description.

```
MCL: LPQ 2

CLASS 2 IS STARTED
          LINKED TO UNSPOOLER 1

ID  STATUS  PRIOR  OWNER  COPIES  TITLE
 6  READY    15    ROOT     10    SPECIFICS
 7  READY    32    TIM       1    PROG1
 5  RUN      32    TIM       1    ORDERS
```

```
MCL: LPQ

UNSPOOLER 1

ID  STATUS  PRIOR  OWNER  COPIES  TITLE
 6  READY    15    ROOT     10    SPECIFICS
 7  READY    32    TIM       1    PROG1
 5  RUN      32    TIM       1    ORDERS

UNSPOOLER 2

ID  STATUS  PRIOR  OWNER  COPIES  TITLE
10  READY    32    TIM       2    ADVICE
11  READY    32    TIM       1    TRACE1
 9  RUN      32    TIM       1    INVOICES
```

LPR prints the contents of a byte-stream file. The specified file or files form a printing job, which may be directly sent to one of the system printers or appended to one of the spool classes.



where:

file-name is the name or the path name of the byte-stream file to be printed.

sysprinter specifies the system printer to be used, in the range 1 to 8, corresponding to the printers SYSPRT1 to SYSPRT8.

class specifies the spooler class to which the print job is appended, in the range 1 to 16.

copies specifies the number of copies to be printed, with a default of 1.

title specifies the print job title, as a character string of length 40 at most. The string may contain blanks only if contained in quotes. If no name is given, the job has no title.

ST specifies the status of the print job, where:

R indicates READY
H indicates HOLD.

"N is the no-copy option, but also removes the "new-page" command (see "Characteristics").

"D is the option specfiying that the file must be deleted after being printed (to be used with the **"N** option).

"A is the option for requesting alignment of paper on the printer.

"G is the option for appending prints on the global SPOOL system.

Direct Printing

If directed to the system printer, the job is printed only if the system printer is not being used by the spooler, and it is not occupied with printing another job.

If no further parameters are specified, the printing job is allocated to the SYSPRT1 system printer.

The command cannot be redirected.

SPOOL Printing

In this case, the following message is displayed when the command is executed:

SPOOL JOB nn

which signifies that the job has been queued in the specified spool class and supplies number "nn" which identifies the job in the system. This number is shown under "JOBID" in the LPQ and SPOOL commands, and has the same meaning for both.

Normally, a copy of the specified file is made and then printed. If the no-copy option is used, the original file is printed. This file may be changed between the LPR command being given and the file being printed.

Characteristics

1. This command has strong connections with the SPOOL system (see Chapter 2, "SPOOL System").

2. If option "**A** is specified in the command, a request for paper alignment on the printer is sent to the global master workstation. Use of the reserved MTA command allows two-way communication with the local master workstation until alignment is obtained. If the print job refers to the workstation printer, this dialogue takes place using the video of the workstation to which the printer is connected.

3. If the file specified in the command is an "alias" file, the corresponding "aliased" file is printed.

4. In a multi-file LPR print, unless "**N** is specified, a "new-page" command is automatically sent at the end of each file printed, and each file print begins on a new page. The no-copy option "**N** also has the effect of omitting the "new-page" commands, so that each file print begins on the page where the previous file print ended.

5. Byte-stream print files must have at least one line-feed character (LF) in every 768 characters.

Examples

1. LPR FILEA FILEB PRT=2

2. LPR /IPL/USR/MARILYN/INVOICE

3. LPR FILE1 FILE2 FILE3 SP=1 CP=3 TL=STORE ST=H

Returns information about files, directories, or volumes specified in the command.

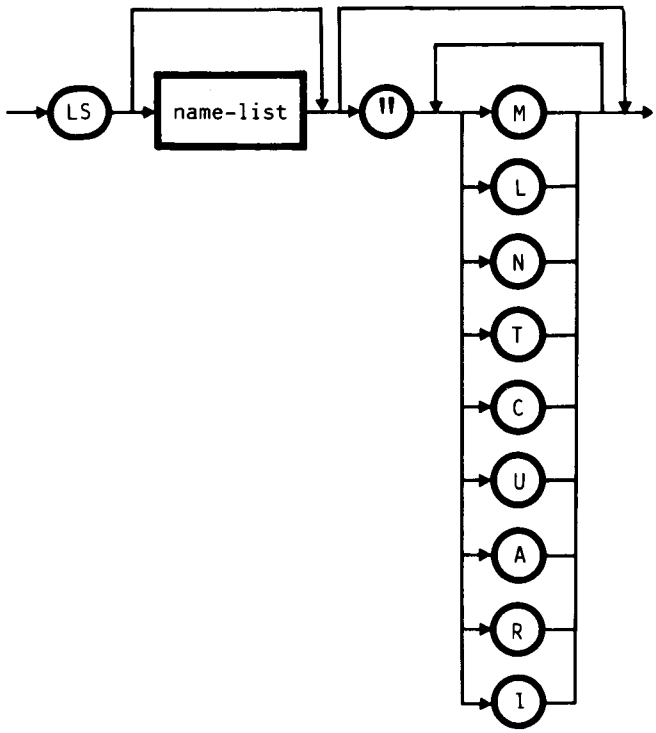
For a volume, LS displays the file names of the directories it contains.

For a file, LS displays its name followed by any information requested with the options.

If the command has no parameters the contents of the current working directory are displayed.

If no option is specified, four file names per line are displayed in no specific order (two file names for a small video).

At the end of each subdirectory display, the number of files contained is shown.



where:

name-list is the list of the names of volumes, directories, or files about which information is to be displayed.

"M is an option requesting the display of the file type and size next to its name (alternative to option **"L**).

"L is an option requesting a complete listing of the file system objects with the following information (but not the file data):

- name
- type
- dimension in bytes
- user code (default value = 0)
- creation date (**"LC**) or of last modification (**"LU**)
- access rights

As an alternative to the option **"M** , this gives the total number of bytes used by all the objects listed.

"N is the option requesting display of the file names in alphabetical order (alternative to option **"T**).

"T is the option requesting display of the file names according to the last update date (must be used with option **"L** and as an alternative to option **"N**).

"C is the option used with option **"L** to display the date the file was created, rather than when it was last updated.

"U is the option used with option **"L** to display the date the file was last accessed, rather than when it was last modified.

"A is an option requesting display of information on the system files (eg: "." and "..") or auxiliary files (containing $), which are not usually displayed.

"R is an option requesting display of the contents of a directory and all its sub-directories.

"I is an option requesting the display, for each of the file system objects present in the specified directory, of the identities of the owner and user group, and the access rights.

Characteristics

1. If two or more options are specified that are alternatives to each other (for example, **"LM**), only the first is executed.

2. The LS command requires access rights to execute and read the directories.

3. For the **"R** option, access rights to execute and read the sub-directories are also required.

4. For the **"N** and **"T** options, access rights to write and append are required on the directory "/".

Examples

1. The command **LS DPC OWS SP** displays information of the following type:

```
DPC:
CMD       COMMON      GRAPHICS      ADMS.
DMS       INITDD
TOTAL 6 FILES

OWS:
OWS:      NAME NOT FOUND

SP:
SPEPA     CONF
TOTAL 2 FILES
```

2. The command **LS MYDIR "M** displays information of the following type:

```
PROVA          VOL        100352
CVP1653        BST           329
P              BST           287
K              KEY             0
PIPPO          POS             0
OUTPUT         BST           880
TOTAL 6 FILES
```

3. The command **LS DPC1 "L** displays information of the following type:

```
CMD     DIR    1818     0  JUN  23    16 : 02 : 28    1986  RAWX  RAWX  RAWX
 :       :      :           :
ADMS    BST    89088    0  JUN  23    16 : 07 : 25    1986  RAWX  RAWX  RAWX
OLINK   PDR    2808     0  JUN  23    16 : 09 : 26    1986  RAWX  RAWX  RAWX
PASCAL  VOL    984576   0  JUN  24    16 : 08 : 30    1986  RAWX  RAWX  RAWX
TOTAL 1563608 BYTES ON 12 FILES
```

4. The command **LS DPC2 "R** displays information of the following type:

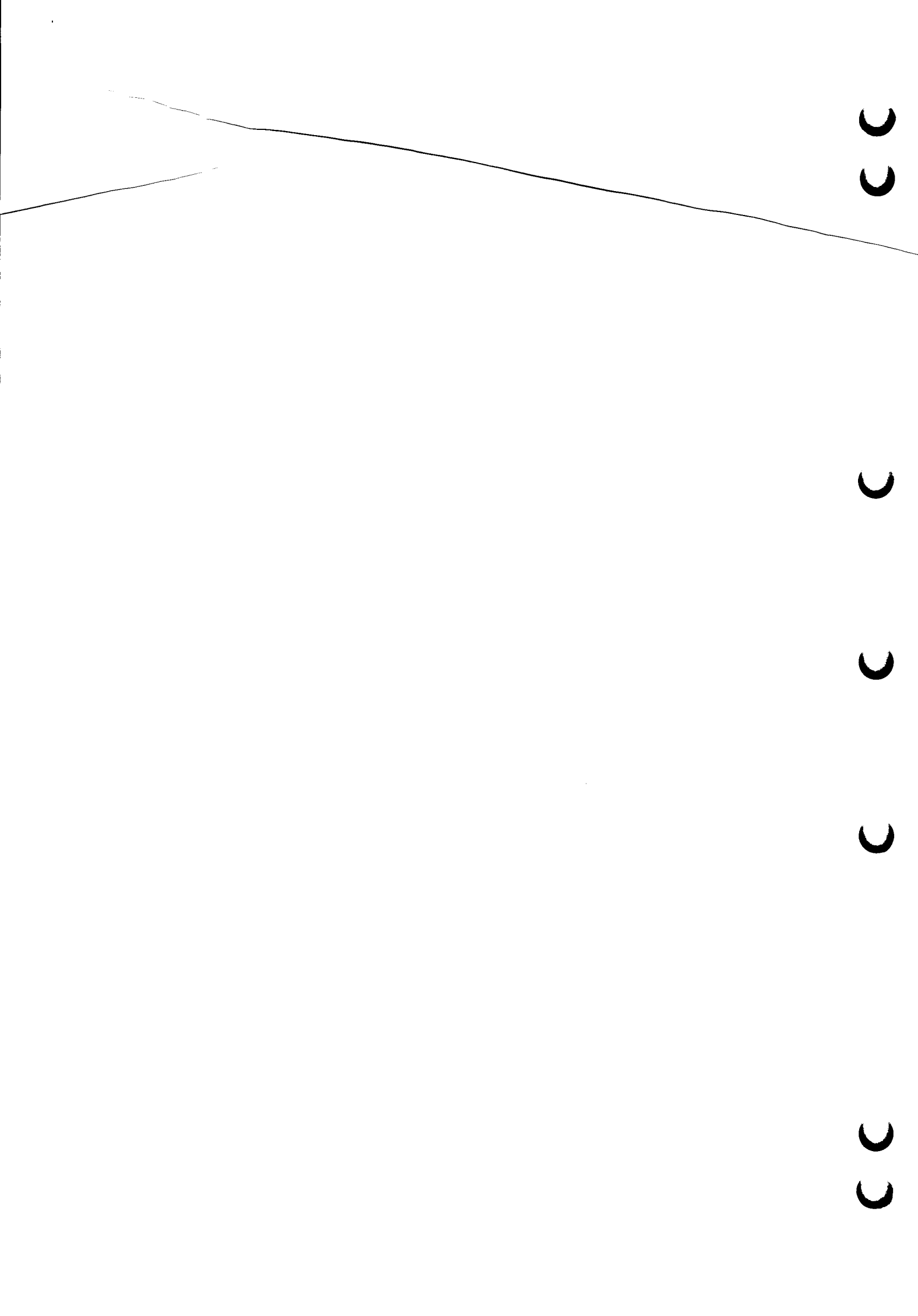
```
                CMD         COMMON        GRAPHICS      ADMS
                TOTAL 4 FILES

                CMD:
                FCU         FCUTAB        TCU           CLEARDIR
                FILETAR
                TOTAL 5 FILES

                CMD/FCU
                MAIN        FCUTABLE      FCU
                TOTAL 3 FILES
```

5. The command **LS DPC3 "LNCA** displays information of the following type:

```
DPC:      DIR     558    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
ADMS      BST   19812    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
BASIC     BST   63504    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
BITMAP    BST     256    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
CALL      BST   26624    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
 :         :      :       :
 :         :      :       :
SYSTEM    BST   33347    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX

maxi      BST   77016    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
merge     BST    1024    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
minisoft  BST    6144    0  JAN  01    01 : 41 : 33    1987  RAWX  RAWX  RAWX
TOTAL 886145 BYTES ON 30 FILES
```

6. The command LS DPC4 "I displays information of the following type:

```
.INIT                 USER1       OTHER       RAWX  RAWX  RAWX
NOSE                  USER1       OTHER       RAWX  RAXW  RAWX
BST                   USER1       OTHER       RAWX  RAWX  RAWX
FILE                  ROOT        SYSTEM      RAWX  RAWX  R--X
DIRE                  USER1       OTHER       R--X  RAWX  RAWX
TOTAL  5  FILES
```



Displays the name of the machine on the screen in the following format:

MACHINE NAME: NnMm

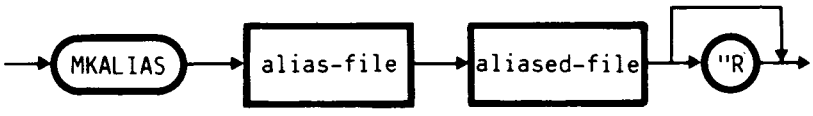
where "n" is a number uniquely identifying the network to which the machine is connected and "m" identifires the machine itself.

The name is also loaded in the MCL variable %RET.

The command has no parameters.

MKALIAS creates a file (alias) which contains the name of another (aliased) file.

The alias file points to the other file, so that if certain operations are requested on the alias file, they are performed on the aliased file. (See the table "Command Operation in an Alias Situation" below.)



where:

alias-file is the name of the alias file to be created, possibly remotely. This may be specified as either a name relative to the working directory, or a complete path name.

aliased-file is the complete path name of the local file which is to be aliased. This file need not exist when the MKALIAS command is issued, but problems may be avoided if it does. When used, the aliased file and the alias file must exist on the same machine.

"R is an option for replacing a previously created alias file with the one being created.

Characteristics

1. MKALIAS functions non-interactively.

2. REMOVE can delete an alias file.

3. All the errors that may occur when creating the alias file are displayed as follows:

   CREATING file name: error message

   where "error message" gives the nature of the error (see appendix A).

4. Access rights for execution, append, and writing on the directory of the alias file are required to execute this command.

5. Shell commands vary in their response to "aliasing" (see table below).

BATCH	-	
BMQ	-	
BUILDER	-	operates on the aliased file
CALLMWS	-	operates on the aliased file for DATA = alias.
CHTYPE	-	error - INVALID OPERATION
CLEARDIR	-	error - INVALID OPERATION
CMPK	-	error - INVALID OPERATION
COMPACT	-	operates on the aliased file
CONVERT	-	error - INVALID FILE TYPE
COPY	-	alias file managed like a byte-stream file
CPTREE	-	operates on the aliased file
CSIZE	-	operates on the aliased file
DATAC	-	operates on the aliased file
DATACONF	-	does not use alias files error - INVALID OPERATION
DELINDEX	-	operates on the aliased file
DIFF	-	operates on the aliased file
DISKCHECK	-	operates on the aliased file
DISKEDIT	-	operates on the aliased file
DMPRINT	-	operates on the aliased file
EXIST	-	verifies that the alias exists
FLD	-	operates on the aliased file
HEXED	-	operates on the aliased file
INSTALLQM	-	operates on the aliased file
KEYCHECK	-	operates on the aliased file
LIST	-	operates on the aliased file
LISTMSG	-	operates on the aliased file
LPR	-	operates on the aliased file
LS	-	operates on the aliased file
M5LDUMP	-	operates on the aliased file
MKALIAS	-	sets-up an alias file
MKBOOT	-	not to be used with an alias file error - INVALID OPERATION
MKCEM	-	not to be used with an alias file error - INVALID OPERATION
MKGLOB	-	operates on the aliased file
MKINDEX	-	operates on the aliased file
MKLOGIN	-	error - INVALID OPERATION
MKVOL	-	operates on the aliased file
MNT	-	error - NAME ALREADY EXISTS
MORE	-	operates on the aliased file
PACK	-	not to be used on an alias file error - INVALID OPERATION
PARSER	-	operates on the aliased file
PRY	-	operates on the alias file
PVOL	-	error - INVALID OPERATION
REMOVE	-	removes alias file
RENAME	-	renames alias file
REPLOG	-	operates on the aliased file
REPTRA	-	operates on the aliased file

Tab. 1 Command Operation in an Alias Situation (cont.)

RP	-	operates on the aliased file
SETBUF	-	operates on the aliased file
SETIDEN	-	operates on the alias file
SETINFO	-	operates on the aliased file
SETMODE	-	operates on the alias file
SHALIAS	-	operates ONLY on the alias file
SHDIR	-	operates on the aliased file
SYSERM	-	not to be used with an alias file
		error - INVALID OPERATION
UNMNT	-	error - INVALID PARAMETERS
UNPACK	-	error - BAD FILENAME
		error - INVALID OPERATION
VCOMPACT	-	error - INVALID OPERATION
VOLGC	-	error - INVALID OPERATION
VOLSR	-	operates on the aliased file
WHEREIS	-	not to be used with alias file
WSTAT	-	operates on the aliased file

Tab. 1 Command Operation in an Alias Situation

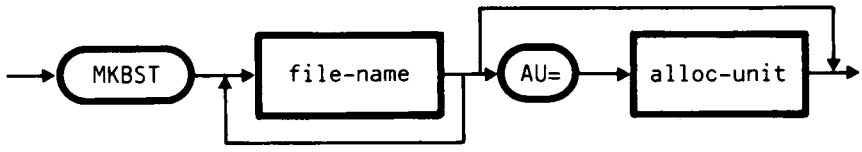
Example:

A file called "NUMBER" has the following full path name: "/COUNT/ONE/TWO/THREE/FOUR/NUMBER". A procedure is being written which makes frequent reference to this file, and also writes another file "HUNDRED" to the same directory, on the same level.

Use of MKALIAS can avoid the repeated keying-in of this path name in the procedure, by creating an alias file named "/A/B" whose only contents are the full path name "/COUNT/ONE/TWO/THREE/FOUR/NUMBER". The procedure can then refer to it more conveniently by quoting the short alias file name, "/A/B".

Moreover, if the procedure is then required to refer to a different directory or file, it is only necessary to change the contents of the alias file "/A/B", instead of amending the procedure at all the points where reference is made to the file.

Creates one or more byte-stream files.



where:

file-name is the name or path name of the byte-stream file.

alloc-unit is the file allocation unit and must be in the range 1 to $2^{31}-1$. The value is rounded to the next highest multiple of 512. If the value specified does not correspond to one of the values defined by the system, it will be rounded up so as to be consistent with the allocation strategy. If this parameter is not specified, a default value of 512 is assumed.

Characteristics

1. The allocation unit optional parameter applies to all the files specified in the command line. It may be modified using the command SETINFO, after the file has been created.

2. All errors which can occur during the execution of MKBST are displayed as follows:

   CREATING path name: error message

   where "path name" is the name or path name of the file system object which has caused the error, and "error message" specifies the nature of the error (see Appendix A).

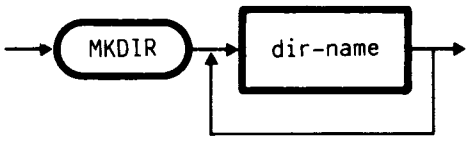
3. Access rights to write or append on the father directory of the file to create are required to execute this command.

Examples

1. MKBST PROGRAM AU=10000

2. SET %FILE := '/IPL/USR/FRANK/EDIT'
   MKBST %FILE&'PROG1'

Creates one or more directories.



where:

dir-name is the name or path name of the directory to be created.

Characteristics

Access rights for writing or append on the father directory of the directory to create are required to execute this command.

Examples

1. MKDIR COBOL/PROGS

   creates the PROGS directory under the COBOL directory.

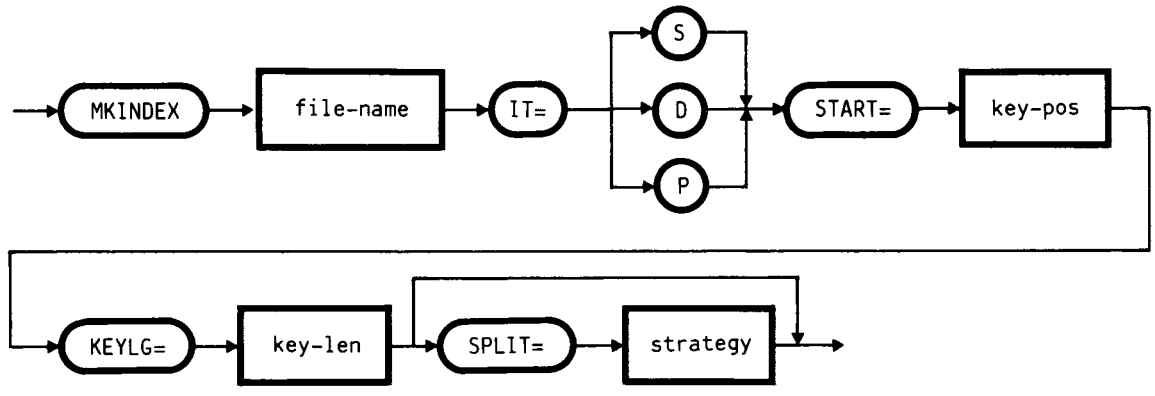
2. MKDIR INVOICE

   creates the INVOICE directory under the current working directory.

Creates an index (that is, defines a new key) in a keyed file. A maximum of 5 secondary indices may be created.

This new secondary index may hold "unique keys", where each key corresponds to only one file record, or "duplicate keys", where each key may correspond to more than one record in a file. The new key must be defined within the record.

In addition to specifying the key position in the record and the key length, it is also possible to determine the split strategy for the B*-tree page division.



where:

file-name is the name or path name of the keyed file.

IT is the index type, which may be one of:

S secondary index with "unique key"

D secondary index with "duplicate key"

P primary index with "unique key".

key-pos is the offset of the first byte of the new key within the record. The minimum value is 0, which indicates the start of the record.

key-len is the key length in bytes, which must be in the range 1-100.

strategy is the strategy for dividing the B*-tree pages. Permitted values are 50, 90, and 100. If this parameter is not specified, the default value 50 is assumed.

Characteristics

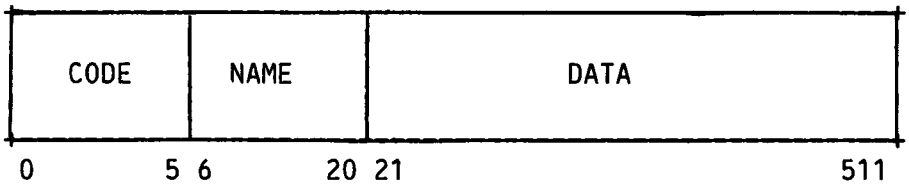
1. The value assigned by the split strategy parameter can be modified using the SETINFO command after the index has been created.

2. To execute MKINDEX access rights for writing or append on the father directory are required. Use of this command is reserved to the primary owner of the file, and the system administrator.

3. When creating the secondary index, the file system assigns to it an identifying index number: this is the next available integer in the list of the indices of the keyed file, including the primary index.

Example

The records of the keyed file CLIENTS have the following format:



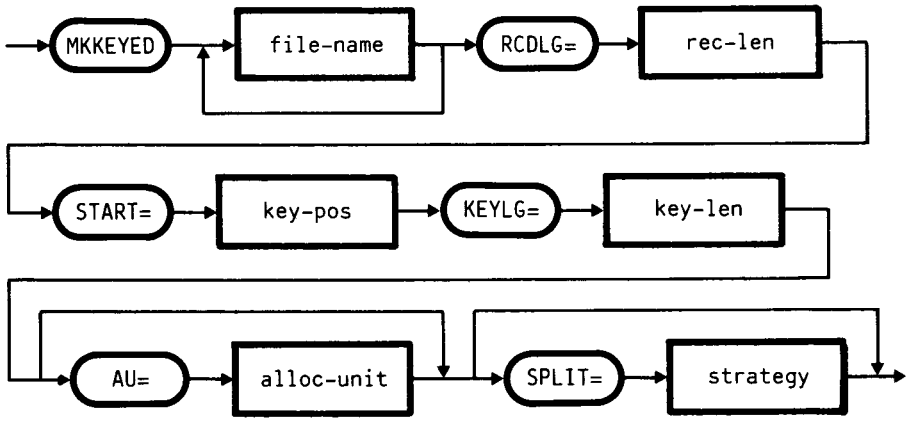
Bytes 0-5 contain the client's code, bytes 6-20 contain the client's surname.

To construct a secondary index with duplicate keys containing the surnames of the clients, the following command must be entered:

Creates one or more keyed files. The key can be defined inside or outside the record, and is defined as the primary key.

Besides parameters which specify the record length, the position of the key relative to the record and the length of the key, there are also parameters to specify the allocation unit and the split strategy of the B*-tree pages.

The specified parameters apply to all the files named in the command.



where:

file-name is the name or path name of the keyed file.

rec-len is the record length in bytes.

key-pos is the offset of the first byte of the key within the record, where the record starts at offset 0; it may have a negative value in which case the key is defined outside the record.

key-len is the key length in bytes, which must be in the range 1-100.

alloc-unit is the file allocation unit and must be in the range 1 to $2^{31}-1$. If this optional parameter is not specified, the default value 512 is assumed.

strategy is the strategy for dividing the B*-tree pages. Permitted values are 50, 90, and 100. If this parameter is not specified, the default value 50 is assumed.

Characteristics

1. The values assigned by the allocation unit and split strategy parameters can be modified using the command SETINFO after the file has been created.

2. All the errors which can occur during the execution of MKKEYED are displayed as follows:

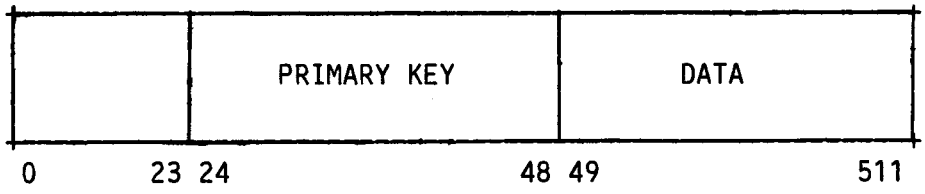
   CREATING path name: error message

   where "path name" is the name or path name of the file system object which has caused the error, and "error message" specifies the nature of the error (see Appendix A).

3. Access rights for execution and append or writing on the father directory of the file to create are required to execute this command.

Examples

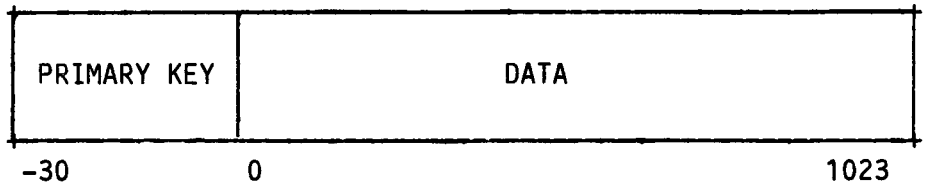
1. To create a keyed file named INVOICES, whose records have the following format:



   The command is as follows:

   MKKEYED INVOICES RCDLG=512 START=24 KEYLG=25

2. To create a keyed file named CLIENTS, whose records have the following format:



   The command is as follows:

   MKKEYED CLIENTS RCDLG=1024 START=-1 KEYLG=30

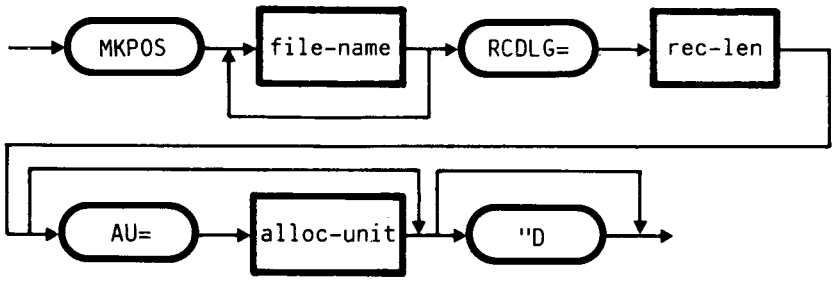
Creates one or more positional files which may be:

- "with record deletion not permitted"

- "with record deletion permitted".

There is no need for a special option to create a file with record deletion not permitted. In the second case the option **"D** must be specified. It is possible to specify an allocation unit, in addition to the record length.

The specified parameters apply to all the files named in the command.



---

where:

file-name is the name or path name of the file.

rec-len is the record length in bytes. Permitted values are in the range 1 to 32767.

alloc-unit is the file allocation unit and must be in the range 1 to $2^{31}-1$. If this optional parameter is not specified, the default value 512 is assumed.

"D requests the creation of a positional file with record deletion permitted.

Characteristics

1. The value assigned by the allocation unit parameter can be modified using the SETINFO command after the file has been created.

2. The positional files with record deletion not permitted and with record deletion permitted are referred to in the system as "positional no deletion" and "positional deletion" respectively.

3. All the errors which can occur during the execution of MKPOS are displayed as follows:

   CREATING path name: error message

   where "path name" is the name or path name of the file system object which has caused the error, and "error message" specifies the nature of the error (see Appendix A).

4. Access rights to append or write on the father directory are required to execute this command.

Examples

1. MKPOS FILE1 FILE2 RCDLG=256

2. MKPOS FRANK/ALFA/FILE3 RCDLG=256 AU=512 "D

The following functions are provided by MKVOL:

- creation of a new logical volume
- allocation of space on hard disc for total memory dumps.

The command can operate both in interactive and non interactive mode for both operations.

CREATING A VOLUME

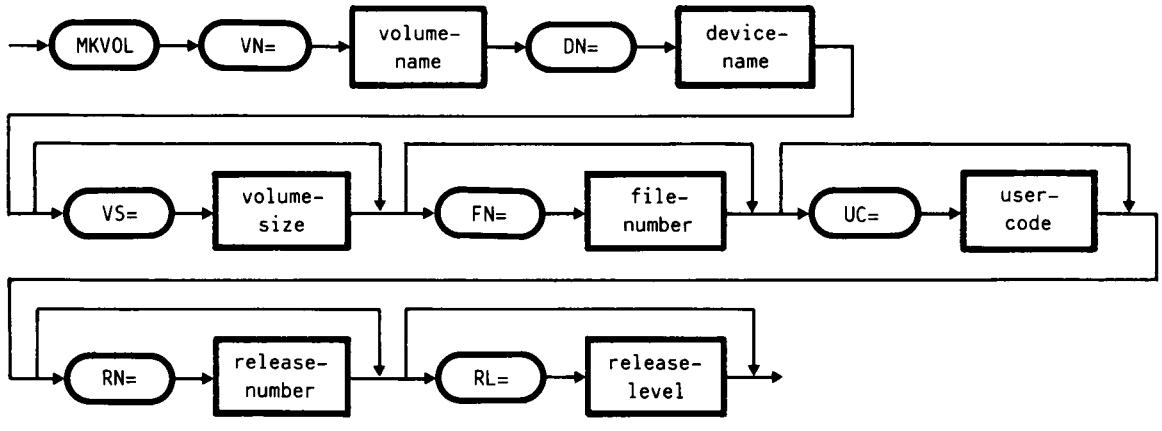
A new volume can be created by specifying the name of a device or a path name.

In the first case, the hard or floppy disc resident on the specified device must first be formatted, and a new volume created on it; this will be the main volume (the command MKENV must be issued first for floppy discs).

In the second case, the new volume is created in an existing volume, under the specified directory.

Before operating on removable volumes, these must be logically connected with the command MNT.

Non-Interactive Mode



where:

volume-name is the name of the volume to create (maximum of 14 characters).

device-name is the name of the device on which the main volume is to be created, or the path name of the existing volume or directory in which the new volume is to be created: valid devices are FL1 to FL4, HD1 to HD8, MF1 to MF8.

volume-size is the size of the volume in bytes: the defauult value is 984576 bytes.

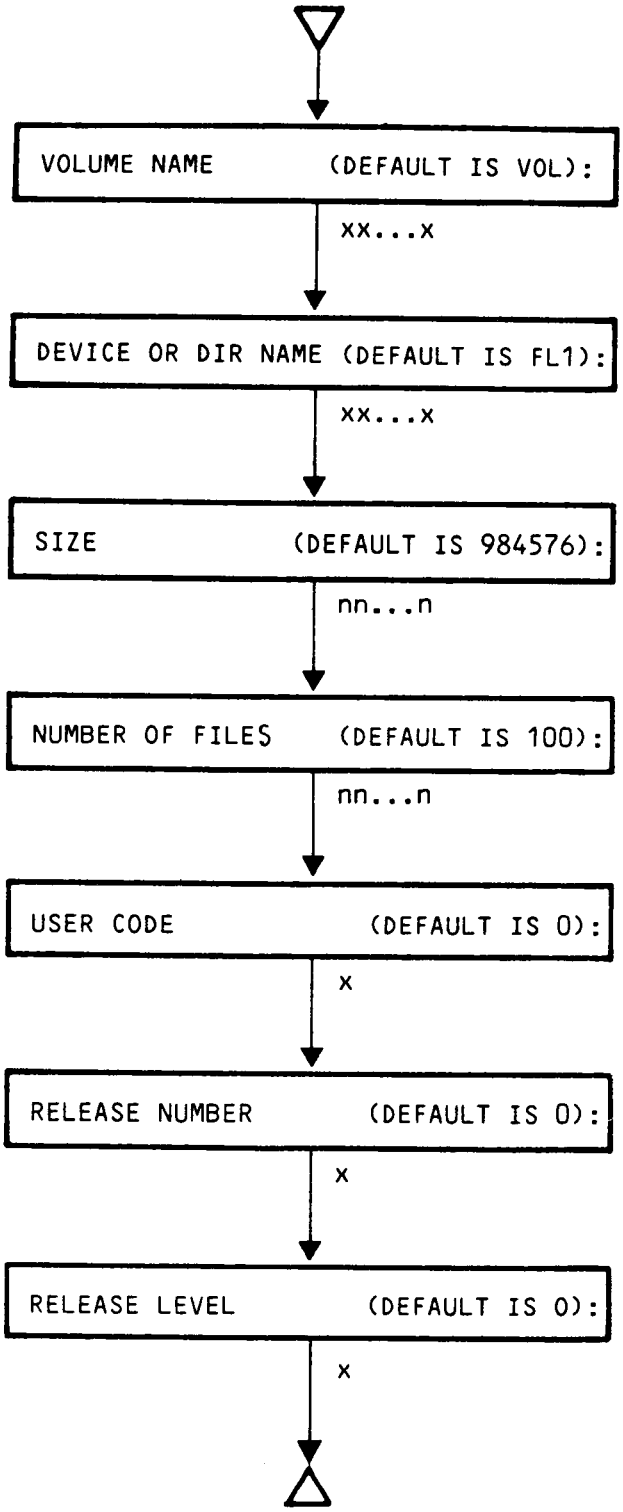
file-number is the maximum number of files that the new volume may contain: the default value is 100.

user-code is the user code used in the volume identifier: the default value is 0.

release-number is the release number also used in the volume identifier: the default value is 0.

release-level is the release level also used in the volume identifier.

Interactive Mode


VOLUME NAME (DEFAULT IS VOL):
xx...x
DEVICE OR DIR NAME (DEFAULT IS FL1):
xx...x
SIZE (DEFAULT IS 984576):
nn...n
NUMBER OF FILES (DEFAULT IS 100):
nn...n
USER CODE (DEFAULT IS 0):
x
RELEASE NUMBER (DEFAULT IS 0):
x
RELEASE LEVEL (DEFAULT IS 0):
x

VOLUME NAME (DEFAULT IS VOL):

Requests the name of the volume to be created.

xx...x Represents the volume name, with a maximum of 14 characters. If the string is longer it will be truncated at the 14th character.

CR Selects the default volume name of VOL.

DEVICE OR DIR NAME (DEFAULT IS FL1):

Requests the name of the device on which the user wants to create the main volume or the path name of the existing volume or directory in which the new volume should be created. Valid devices are FL1 to FL4, HD1 to HD8, MF1 to MF8.

xx...x Represents the name of the device or path name of the existing volume or directory, with a maximum of 14 characters.

CR Selects the default drive FL1.

SIZE (DEFAULT IS 984576):

nn...n Represents the size of the volume expressed in number of bytes.

0, CR Select the default value of 984576 bytes.

NUMBER OF FILES (DEFAULT IS 100):

nn...n Represents the maximum number of files which may be contained within the new volume.

0, CR Select the default value of 100.

USER CODE (DEFAULT IS 0):

Requests the user code, which is used as part of the volume identifier.

x Represents the user code.

CR Selects the default value 0.

RELEASE NUMBER (DEFAULT IS 0):

Requests the release number, which is used as part of the volume identifier.

x Represents the release number.

CR Selects the default value 0.

RELEASE LEVEL (DEFAULT IS 0):

Requests the release level, which is used as part of the volume identifier.

x Represents the release level.

CR Selects the default value 0.

Characteristics

1. The last three parameters (user code, release number, release level) have no other purpose than to serve as volume identifiers.

2. If the volume is to be created in the first level of the system volume, the user must specify the "root directory" path name, i.e. /IPL, when prompted for the "DEVICE OR DIR NAME" parameter.

3. The maximum capacity available to the user on each type of hard disc and floppy disc available is given in Appendix B under "peripherals".

4. Access rights to append or execute and write on the father directory of the volume to create are required to execute this command.

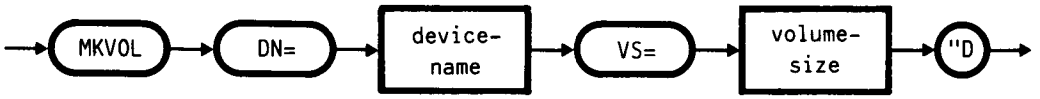
5. Access rights on the new volume, and on the corresponding PDD table and bitmap are for read, write, append and execute for the owner only.

6. Access rights for the "root directory" of the volume created are the default access rights with execution right for all users.

ALLOCATING SPACE ON HARD DISC

This function allows space to be allocated on a hard disc, for instance for the purpose of total memory dumps.

Non Interactive Mode



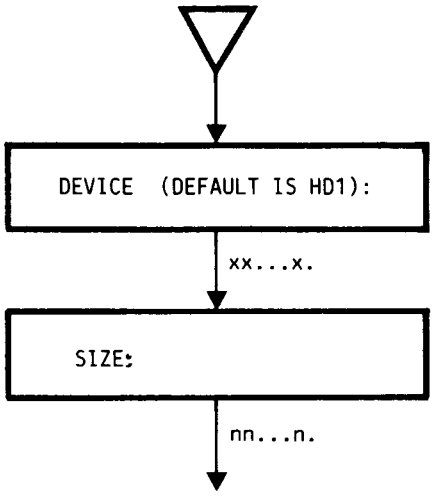
where:

device-name is the name of the device on which space is to be allocated, HD1 or HD2.

volume-size is the size in bytes of the space to allocate on hard disc.

"D is an option requesting space allocation on hard discs for total memory dumps.

Interactive Mode



DEVICE (DEFAULT IS HD1):

Prompts for the name of the device on which the user wants to allocate space for a total memory dump.

xx...x Represents the device name: HD1 to HD8.

CR Selects the default drive HD1.

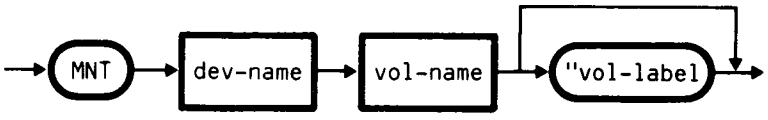
SIZE:

Prompts for the amount of space to allocate on hard disc, expressed in number of bytes.

nn...n Represents the space to allocate in number of bytes.

CR Selects the default value of 984576 bytes.

Logically connects a main volume resident on a specified drive (HD, FD, or MF) to the system. This connection gives the system access to the information contained in the volume.



where:

dev-name is the name of the device on which the volume resides. Valid devices are: HD1-HD8, FL1-FL4, MF1-MF4 (1 Mbyte) and MF5-MF8 (320 Kbyte).

vol-name is the name or path name to be assigned to the volume to be mounted.

vol-label is the physical volume name which should be contained in the disc descriptor of the volume to be mounted. If it contains more than 14 characters, the name is truncated.

The volume must be mounted under the root directory memory volume "/". The user must give the name of an inexistent directory for the volume to be mounted, which will then be created and will identify the mounted volume to the system.

To ensure that the volume label (assigned by the MKVOL command) of the volume to be mounted coincides with that of the actual volume present in the drive, the label parameter must be preceded by double quotes ("). The system compares the specified label with that contained in the disc descriptor: if they agree, the command is executed, otherwise it is aborted.

Characteristics

The following access rights are required to execute this command:

- execute, write and read or append on the directory that is to contain the volume
- read and write on the device containing the volume to be loaded.

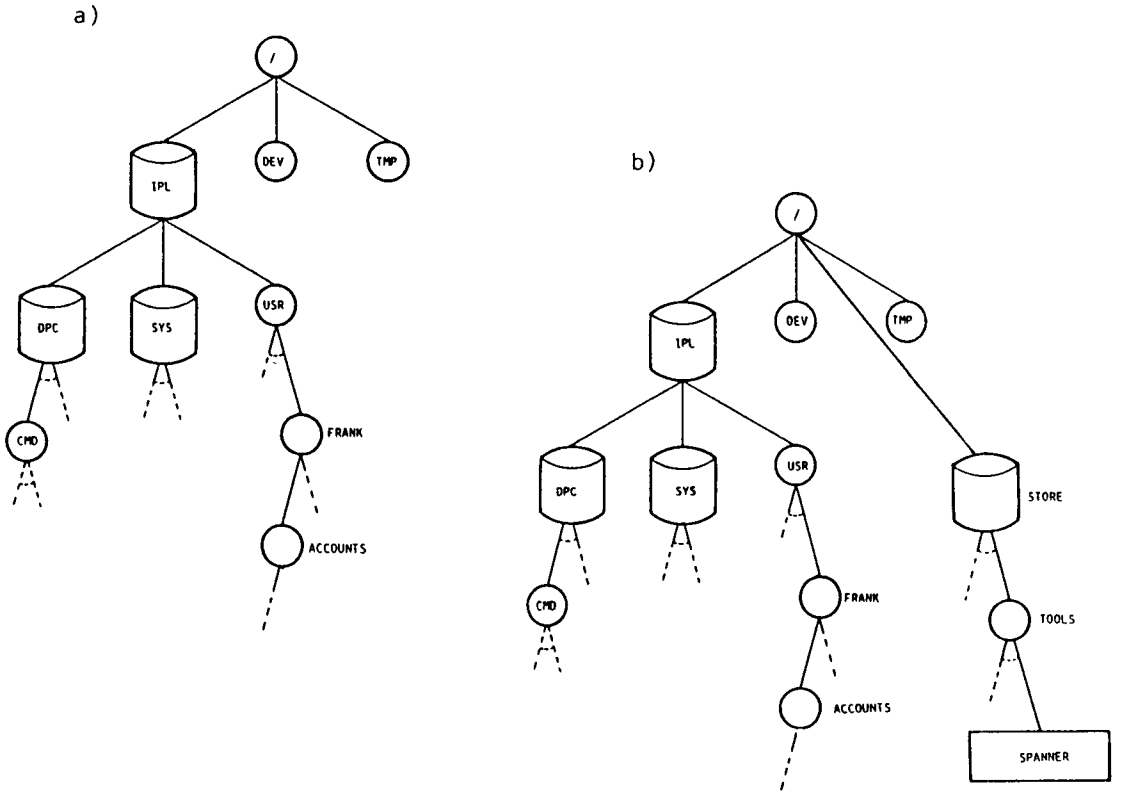


Fig. 3.MNT-1 MNT Example

The structure of the "root directory" (before the MNT operation) is as shown in a) in the diagram.

The command

MNT FL1 /STORE "VOLSTORE

logically connects the volume labelled VOLSTORE to the system. The volume resides on the floppy disc inserted in the floppy drive 1 and is identified to the system by the volume name STORE.

The resulting "root directory" is shown by b).

The SPANNER file in the mounted volume can now be referred to as:

/STORE/TOOLS/SPANNER

Displays the contents of one or more byte-stream files in ASCII characters.

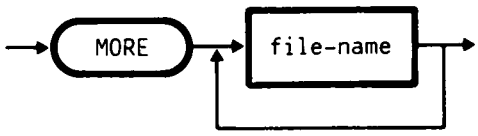
The display starts from the beginning of the first file in the list. After the display of a page of 20 lines (80 characters each), the percentage of the file already displayed is shown, and the user may then choose the next operation to be executed.

Permitted operations are:

- to display the next page of the current file (Ø)
- to display the next line of the current page (carriage return)
- to display the current file name (W)
- to display the current file from the beginning (C)
- to display the next file from the beginning, according to the list of files specified (N)
- to display the preceding file from the beginning, according to the list of files specified (P)
- to display the pages of the file that contain a specific string (S for the first occurrence and carriage return for those that follow)
- to quit the command and return to the Shell environment (Q).

The command also terminates when:

- there are no files before the current one when the user enters **P.**
- there are no files after the current one when the user enters **N.**



where:

file-name is the name or path name of the byte-stream file to be displayed.

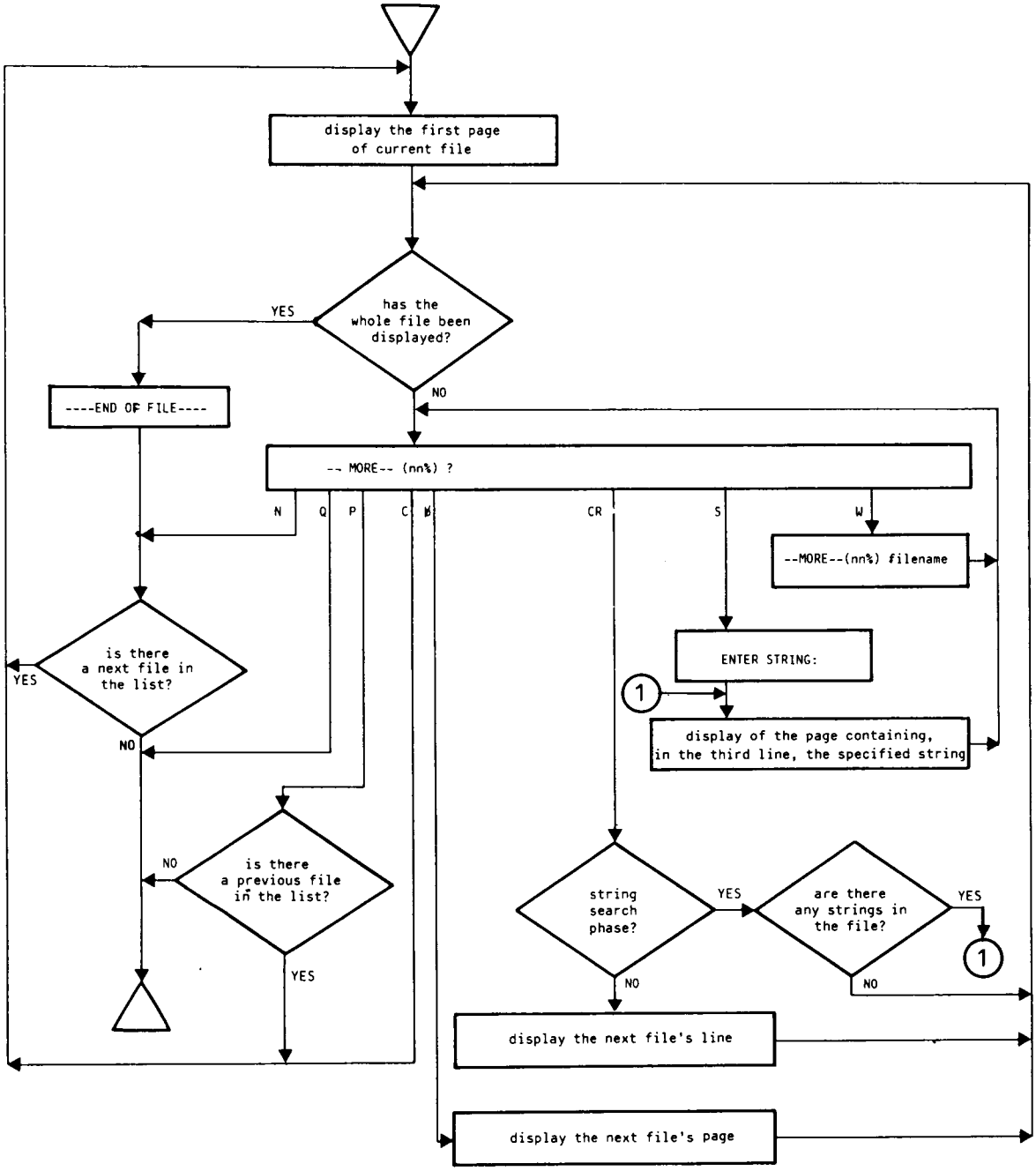
Characteristics

1. All the character parameters can be entered in either upper or lower case.

2. Access rights for execution on the father directory and read on the file to be displayed are required to execute this command.

3. If the string to be searched is not found in the file, the following message is displayed:

--PATTERN NOT FOUND--


display the first page of current file
has the whole file been displayed?
YES
NO
----END OF FILE----
-- MORE-- (nn%) ?
N
Q
P
C
CR
S
W
--MORE--(nn%) filename
is there a next file in the list?
YES
NO
ENTER STRING:
1
display of the page containing, in the third line, the specified string
is there a previous file in the list?
NO
YES
string search phase?
YES
are there any strings in the file?
YES
NO
1
NO
display the next file's line
display the next file's page

Information Messages

-- MORE -- (nn%) ?

This message appears on the last line of the screen page. The user can enter:

N	The first page of the next file in the list of files is displayed. If this was the last file, the command terminates and the system returns to the Shell environment.
P	The first page of the preceding file in the list of files is displayed. If this was the first file, the command terminates and the system returns to the Shell environment.
C	The first page of the current file is displayed again.
SP	The next page is displayed.
CR	The next line is displayed.
W	The name or path name of the current file is displayed.
S	The pages of the file which contain the specified string are displayed.
Q	The command terminates and returns to the Shell environment.

-----END OF FILE-----

This message appears after the last line of the current file has been displayed.

ENTER STRING:

xx...x	The user specifies the string to search for, with a maximum of 40 characters.

Allows the compaction of byte-stream files.

The packed file will have the same name as the original, plus the suffix ".R". The Huffman method (minimum redundancy) is used on a byte-by-byte basis for the compaction, which is achieved in three phases:

1. The character frequency is assessed.

2. A new code is written in which the most frequent characters are represented by a shorter code.

3. The file is re-written using the new code.

The amount of packing to be done depends on the file size and the frequency of characters. Packing on files which are smaller than 3 blocks (1536 bytes) gives no significant advantage. A text file is usually reduced to 60-70% of its original size.

The file can be resumed to its original size using the command UNPACK.

where:

file-list is the list of byte-stream files to be packed; each local name may have a maximum of 12 characters after the final "/".

dirname is the target directory for the packed files; if this is omitted, the working directory is used.

"D is an option requesting the removal of the original file after packing.

"V requests the display of the packing percentage.

Characteristics

1. When a file is being packed the following message is displayed:

   PACKING: file-name

2. After each packing operation successfully completed, the following message is displayed:

   file-name : PACKED

3. If the option "V has been specified in the command line, the following message is displayed:

   nn% COMPRESSION

   where "nn" is the percentage of the file that has been been packed.

4. The following access rights are required to execute the command:

   - execute and write or append on the father directory of the packed file

   - read on the file to pack.

5. The PACK and UNPACK commands can be used when dumping files by means of the utilities FLD, FILETAR, or VOLSR, to make these operations more efficient.

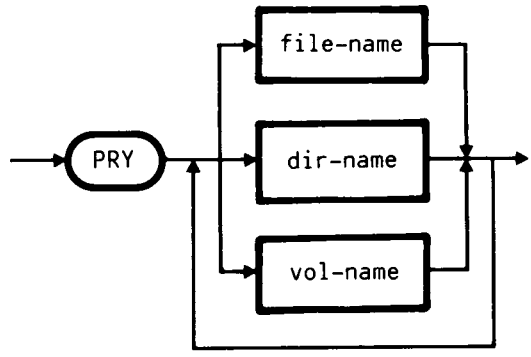
Example

```
      .
      .
      .
PACK FILE22 "V
FLD FUNC=S PATH=/IPL/USR/MARY/FILE22.R DEV=FL1
      .
      .
FLD FUNC=R FPATH=/IPL/USR/JOHN/FILE22.R DEV=FL1
UNPACK FILE22.R
      .
      .
      .
```

Displays information regarding the following file system components:

- files
- directories
- volumes.

Table 1 shows the type of information displayed for each file system object, the meaning of which is given in Table 2.



where:

file-name is the name or path name of the file.

dir-name is the name or path name of the directory.

vol-name is the name or path name of the volume.

Characteristics

1. The command also allows the display of information on removable mounted volumes.
2. The following access rights are required to execute the command:
   - execute on the father directory
   - read on the file system object whose information display is required.

Information displayed	Byte-Stream file	Positional file	Keyed file	Directory	Volume
OWNER	X	X	X	X	X
TYPE	X	X	X	X	X
USER TYPE	X	X	X	X	X
CREATED	X	X	X	X	X
ACCESSED	X	X	X	X	X
MODIFIED	X	X	X	X	X
ALLOC UNIT	X	X	X		X *
SECURITY	X	X	X	X	X
SIZE	X	X	X		X
BUSY SPACE			X	X	
REC LENGTH		X	X		
FIELD NUMBER			X		
INDEX TYPE			X		
KEY START			X		
KEY LENGTH			X		
SPLIT STRATEGY			X		
FREE SPACE					X
FREE GROUPS					X
MAX GROUP					X
REL NUMBER					X
REL LEVEL					X
FILE NUMBER					X
FILES USED					X

Tab. 1 File System Objects and Information Displayed

* ALLOC UNIT has no significance for volumes. It is always displayed as zero.

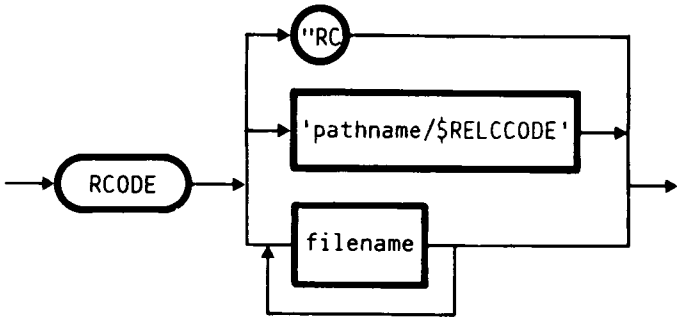
INFORMATION	MEANING
OWNER	Identity of the primary or secondary owner of the file system object.
TYPE	Type of data structure. This may be: ALIAS, BYTE-STREAM, POSITIONAL, KEYED WITH INTERNAL KEY, KEYED WITH EXTERNAL KEY, DIRECTORY, PROGRAM, VOLUME, or FLOPPY DISK 1 MEGA-BYTE.
USER TYPE	Code of the user who created the file system object.
CREATED	Date of creation as recorded by the system.
ACCESSED	Date of last access as recorded by the system.
MODIFIED	Date of last modification as recorded by the system. In the case of volumes, this does not include changes to directories or files within the volume.
ALLOC UNIT	Number of bytes used by the system for each data structure extension.
SECURITY	Access rights of the owner, members of the owner's user group, and other users. The access rights can be any or all of: R = read A = append W = write X = execute
SIZE	Physical size of the file system object expressed in: bytes (byte-stream, device), records (positional file, keyed file with either internal or external key), or blocks (volume).
BUSY SPACE	Logical size of the file system object expressed in: records (keyed or positional file), number of entry points (directory), or blocks (volume).
REC LENGTH	Record length in bytes.

Tab. 2 Meaning of the Information Displayed (cont.)

INFORMATION	MEANING
FIELD NUMBER	Index number. This is assigned in ascending order when the indices are created: 0 = primary index 1-5 = first to fifth secondary index.
INDEX TYPE	Type of index: PM = primary index SU = secondary index with unique keys SD = secondary index with duplicate keys.
KEY START	Key offset with respect to the beginning of the record.
KEY LENGTH	Key length expressed in bytes.
SPLIT STRATEGY	Division strategy of the B*-tree pages. This may be: 50, 90, or 100.
FREE SPACE	The number of blocks (of 512 bytes) available to create or extend files.
FREE GROUPS	The number of groups of contiguous free blocks.
MAX GROUP	The number of blocks in the largest group of free blocks.
REL NUMBER	Release number. This is a volume identifier.
REL LEVEL	Release level. This is a volume identifier.
FILE NUMBER	Maximum number of files in the volume.
FILES USED	The number of simple files that have been created in the volume (the maximum number of files possible is specified in the MKVOL command).

Tab. 2 Meaning of the Information Displayed

Displays either general or specific software release information, according to the request entered.



where:

'pathname/$RELCCODE' (or **"RC**) displays the generalised information contained in a system file storing the main release number, the release level, and any patches included (see examples below).

filename is the name of a specific software module: its main release number, release level number, patch level, environment identifier, and the component code are displayed (see examples below).

Note: Leave a single space between the names, and if a file name begins with '$', enclose it in single inverted commas (').

Examples

1. RCODE 'IPL/SYS/$RELCCODE'

   displays brief information in the following format:

   REL 5.1.1 P1,P2,P3

2. When specific file names are input, such as:

   RCODE '$LOG' '$POS' '$IPL' '$PMM'"

   the display shows:

NAME	MAINR.	REL.LV	PATCH LV	ENV.ID	COMP. CODE
$LOG	5	1	1	01	
$POS	5	1	1	00	
$IPL	5	1			
$PMM	0	0			

Characteristics

1. $PMM has no code.

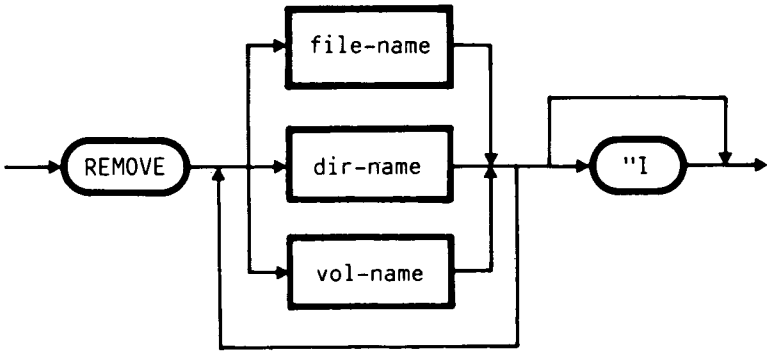
2. The environment identifier field shows either 00 or 01, being an indication of where the software was developed (00 = Olivetti-produced original, 01 = derived from an original using the MOS generation tools).

3. The component code only appears for releases before 5.1, their meaning being explained in the appropriate release guide.

Deletes all the elements specified in a list.

The list can contain names or path names of files, empty directories, or logical volumes. An empty directory does not contain files or sub-directories.

If interactive execution of the command is required, all the names specified in the command are displayed one after another. The user can then confirm their removal by typing **Y** or decide to cancel the request by typing **N**.



where:

file-name is the name or path name of the file to remove.

dir-name is the name or path name of the empty directory to remove.

vol-name is the name or path name of the logical volume to remove.

"I requests interactive execution of the command.

Characteristics

1. It is not possible to remove the directories named "." and "..".

2. If a request has been made for a REMOVE operation concerning an object to which connections have been made, the system immediately removes the name of the object from its directory, therefore stopping further connections to that name, but not does not physically delete the data of that object: this ensures that all the users who have made previous connections can still access the data. This data is only removed when the last connection is released. If this last connection is not released (because of a crash, reset, etc) the disc space occupied by the object that is still connected is not released.

   Disc consistency may be checked using the DISKCHECK command and any wasted space can be recovered by means of the VOLGC command (see the "MOS System Software Maintenance Tools User Guide").

3. It is not possible to perform a REMOVE operation on a volume or directory in which there are objects with active connections; if this is attempted, the command aborts.

4. All the errors which can occur during the execution of REMOVE are displayed as follows:

   REMOVING path name: error message

   where "path name" is the name or path name of the file system object which has caused the error, and "error message" specifies the nature of the error (see Appendix A).

5. Access right in writing on the father directory of the file system object to remove is required to execute the command.

Example

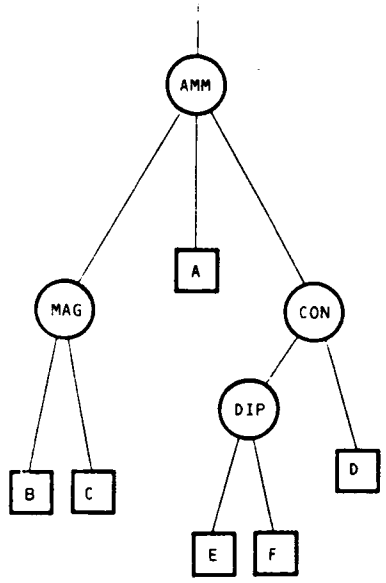
Given the memory structure in a) of the figure of the opposite page, entering the command:

REMOVE AMM/CON/DIP/E AMM/CON/DIP/F

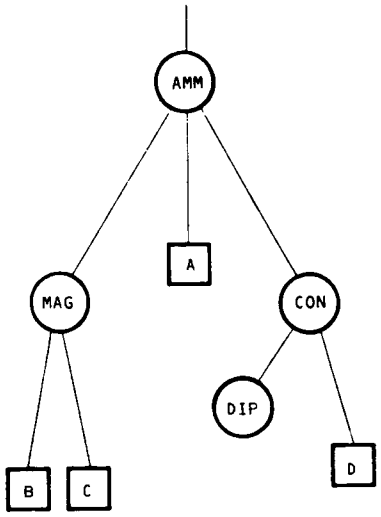
will give the structure b) and entering the command:

REMOVE AMM/CON/DIP

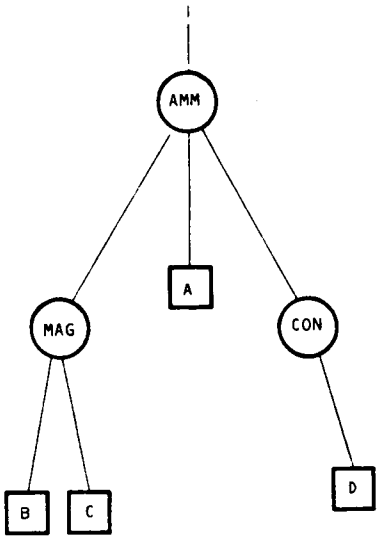
will give the structure c).



a)



b)



c)

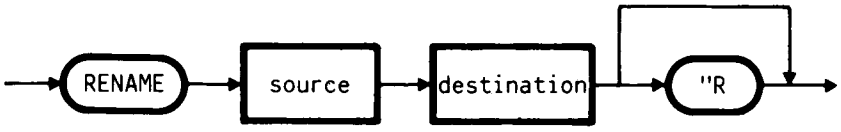
Fig. 3.REMOVE-1 REMOVE Example

Changes the name of a "source" file, directory, or volume to another "destination" name.

The command operates differently depending on whether the "source" and "destination" are files, directories, or volumes. Each possible combination of file, directory, or volume, for both "source" and "destination" is described below.

If a destination file is specified, the name may have a maximum of 12 characters.

The general format of the command is as follows:



where:

source is the name or path name of the "source" file, directory, or volume.

destination is the name or path name of the "destination" file, directory, or volume.

"R is the replace option.

Characteristics

1. As the RENAME command involves a removal operation, see the command REMOVE.

2. In a distributed system "source" and "destination" must reside on the same machine.

3. If source and destination are in the same volume, the source file attributes (access rights and owner identity) are given to the destination file. If however, they are in different volumes, the destination files are given the attributes defined in the default access list of the user performing the command.

4. RENAME cannot be used on the directories "." and "..".

5. The following access rights are required to execute the command:

   - write on the father directory containing the object specified in "source"

   - write or append on the father directory of the object specified in "destination".

RENAMING A FILE

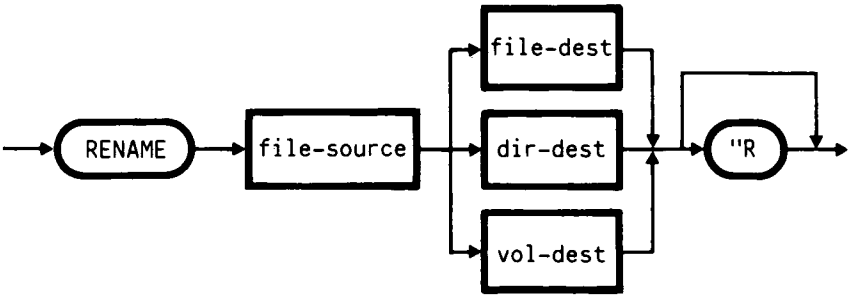
The "source" is a file, the "destination" is:

- a file

  If the destination file does not exist, the source file takes the name of the destination file. If it does exist, it is overwritten by the source file only if the replace option was specified, otherwise the command is aborted.

- an existing directory or volume.

  The source file is removed from its position and inserted in the first level of the destination volume or directory. If the first level of this directory or volume already contains a file system object having the same name as the source file, the source file will overwrite it only if the replace option is specified, otherwise the command is aborted.



where:

file-source is the name or path name of the source file to rename.

file-dest is the name or path name of the destination file.

dir-dest is the name or path name of the destination directory.

vol-dest is the name or path name of the destination volume.

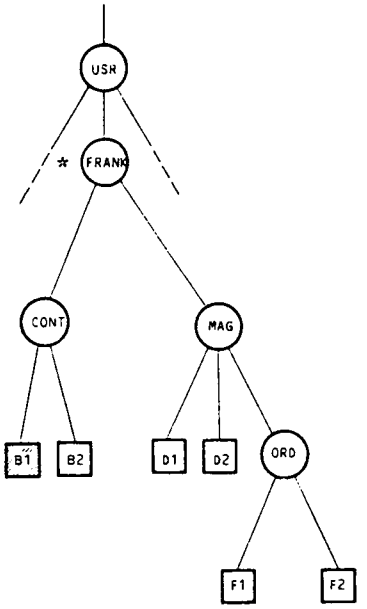
"R is the replace option.

Note: If source and destination are in different volumes and the source file is open, the command aborts.

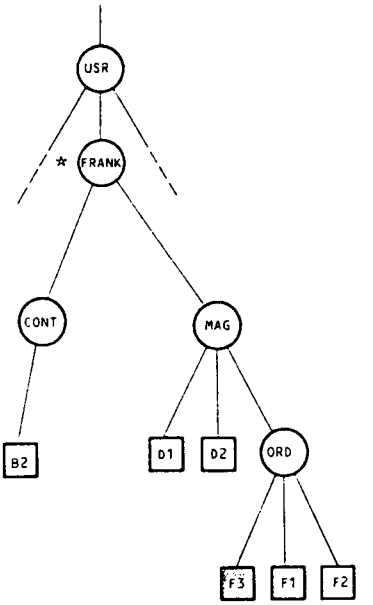
Examples

1. RENAME CONT/B1 MAG/ORD/F3

before



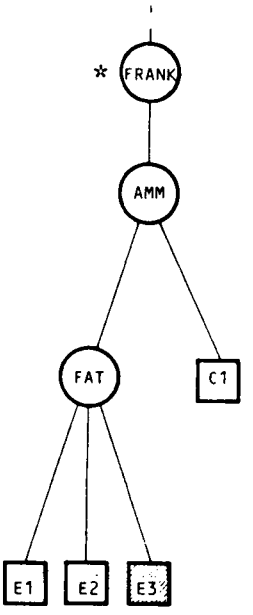
after



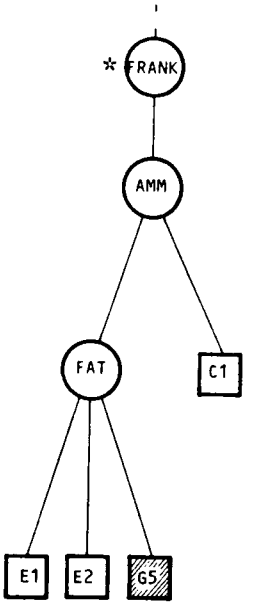
---

2. RENAME AMM/FAT/E3 AMM/FAT/G5

before

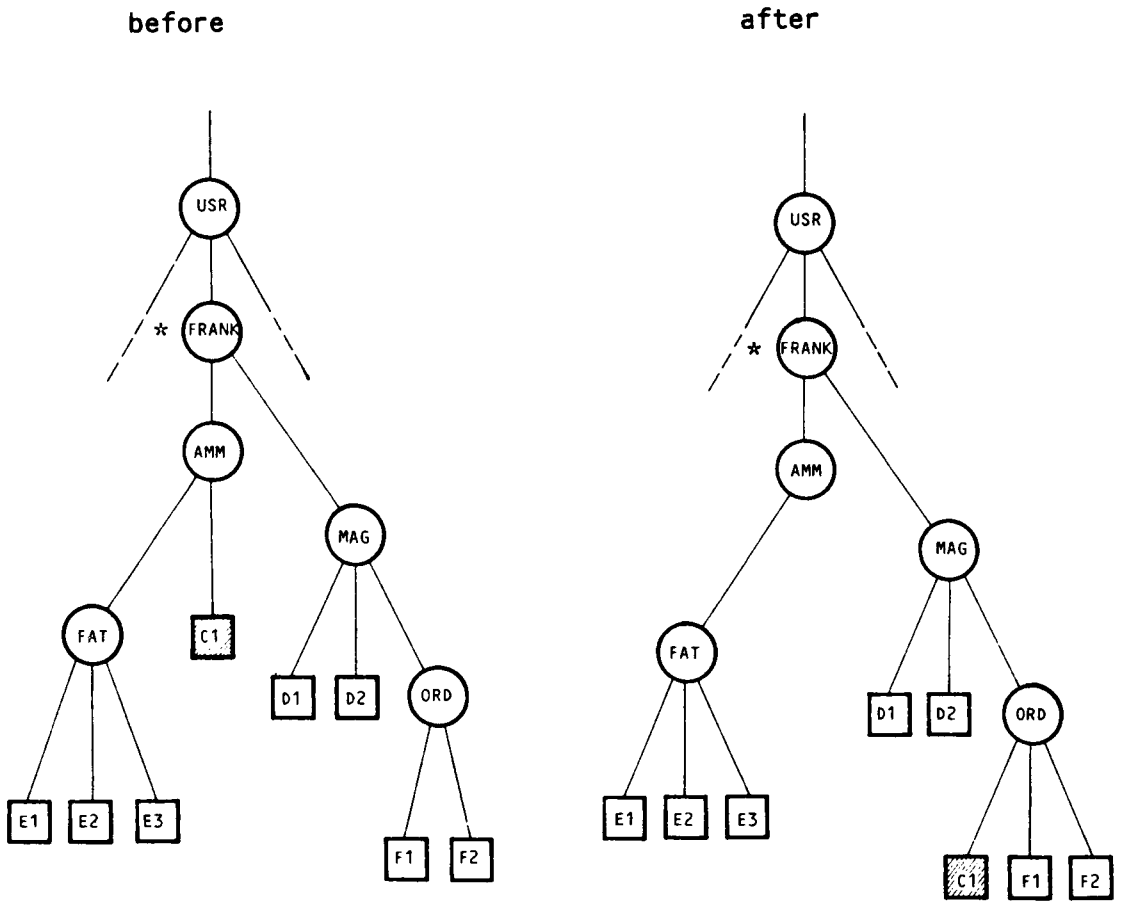


after



---

3. RENAME AMM/C1 MAG/ORD



RENAMING A DIRECTORY

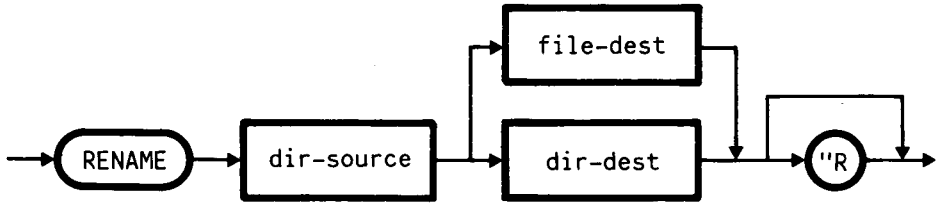
The "source" is a directory and the "destination" is:

- an existing file

  If the replace option is specified, the destination file is removed and the source directory takes its name. The command is aborted if the replace option is not specified.

- a directory.

  If the destination directory does not exist, the source directory takes the destination name. If it exists, the command is aborted even if the replace option is specified.



where:

dir-source is the name or path name of the source directory to rename.

file-dest is the name or path name of the destination file.

dir-dest is the name or path name of the destination directory.

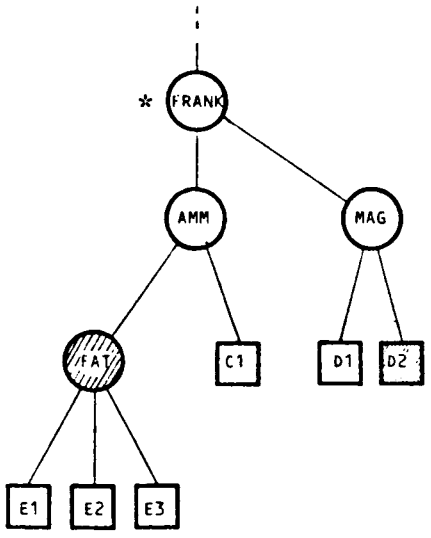
"R is the replace option.

Note: The command is aborted if there are operations in progress on the directory to be renamed.

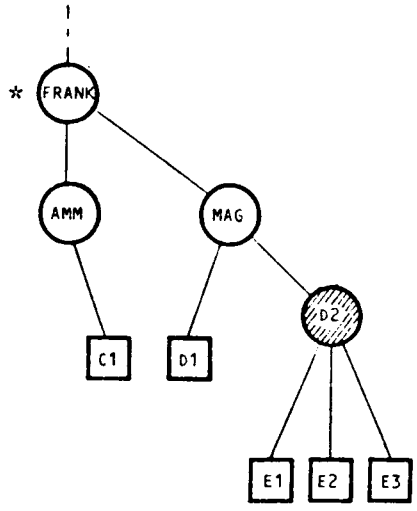
Examples

1. RENAME AMM/FAT MAG/D2 "R

before

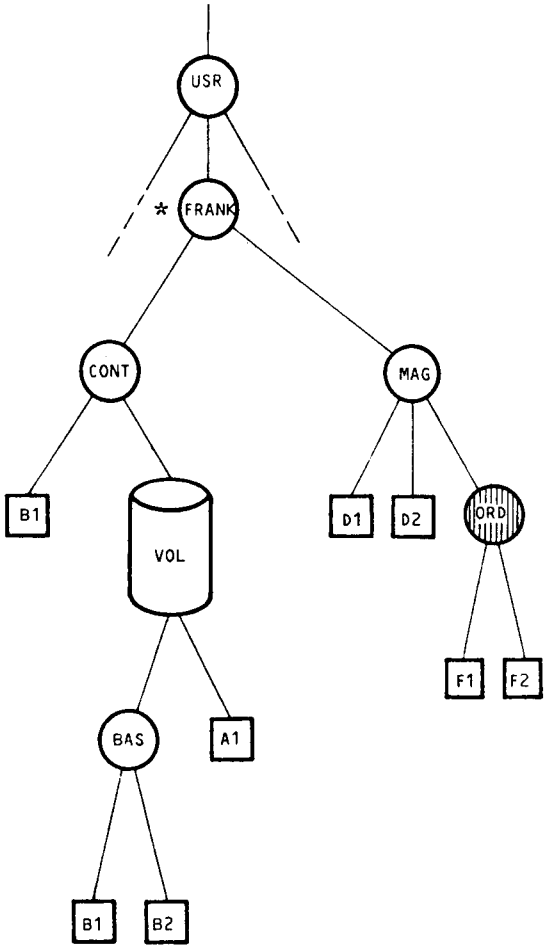


after

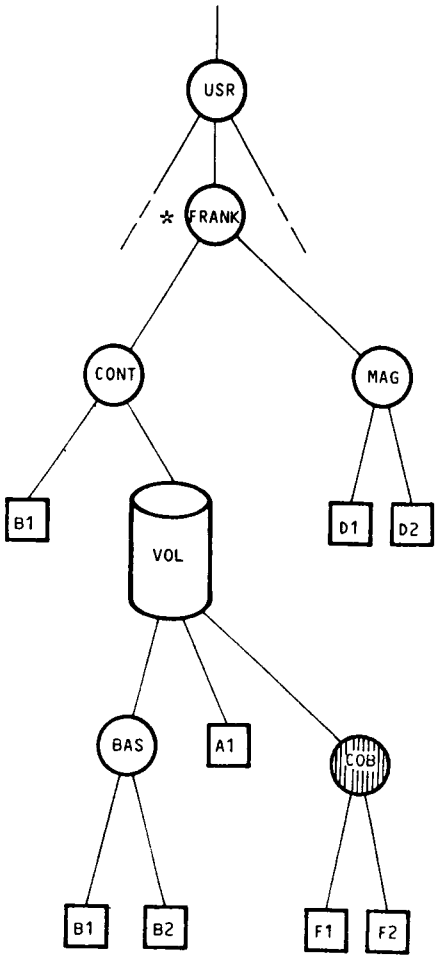


2. RENAME MAG/ORD CONT/VOL/COB

before



after



RENAMING A VOLUME

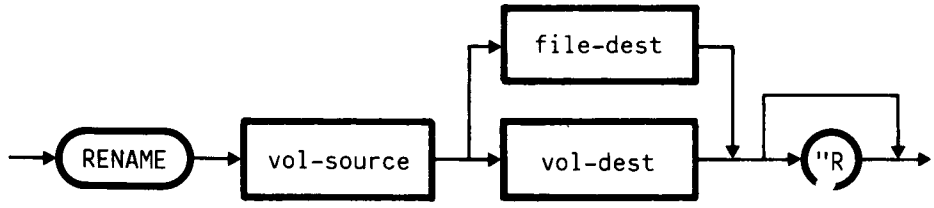
The "source" is a volume and the "destination" is:

- an existing file

  If the replace option is specified, the destination file is removed and the source volume takes its name. The command is aborted if the replace option is not specified.

- a volume.

  If the destination volume does not exist, the source volume takes the destination name. If it does exist, the command is aborted even if the replace option is specified.



---

where:

vol-source is the name or path name of the source volume.

file-dest is the name or path name of the destination file.

vol-dest is the name or path name of the destination volume.

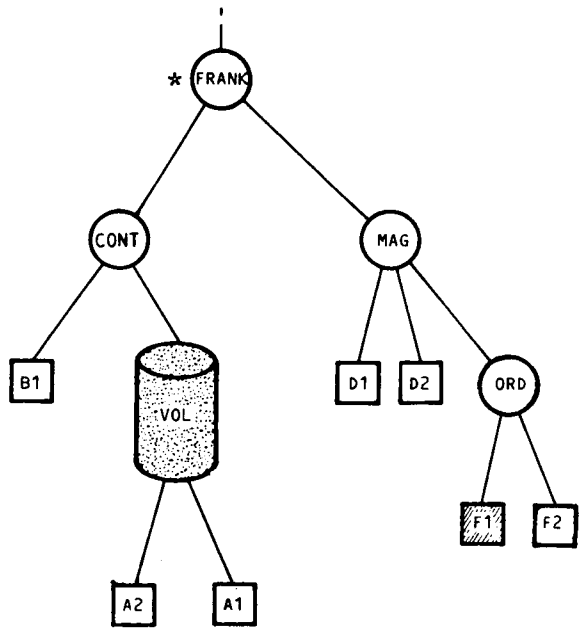
"R is the replace option.

Note: The command is aborted if there are operations in progress on the volume to be renamed.

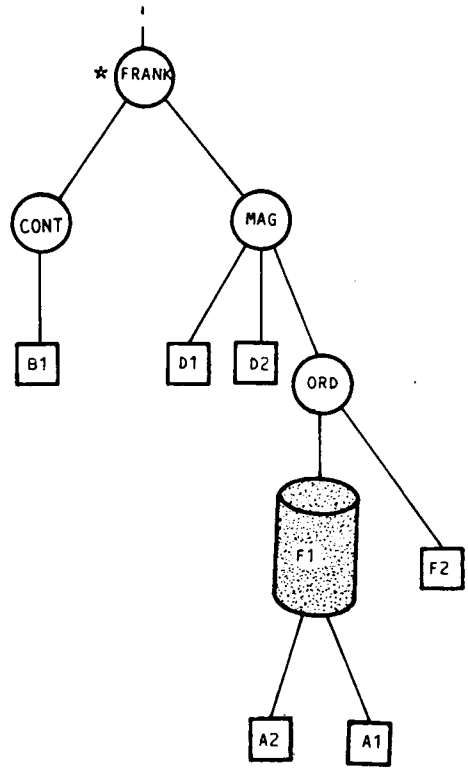
Example

```
RENAME CONT/VOL MAG/ORD/F1 "R
```

before



after



Resumes the execution of a user program that has been previously suspended by the SUSPEND command.

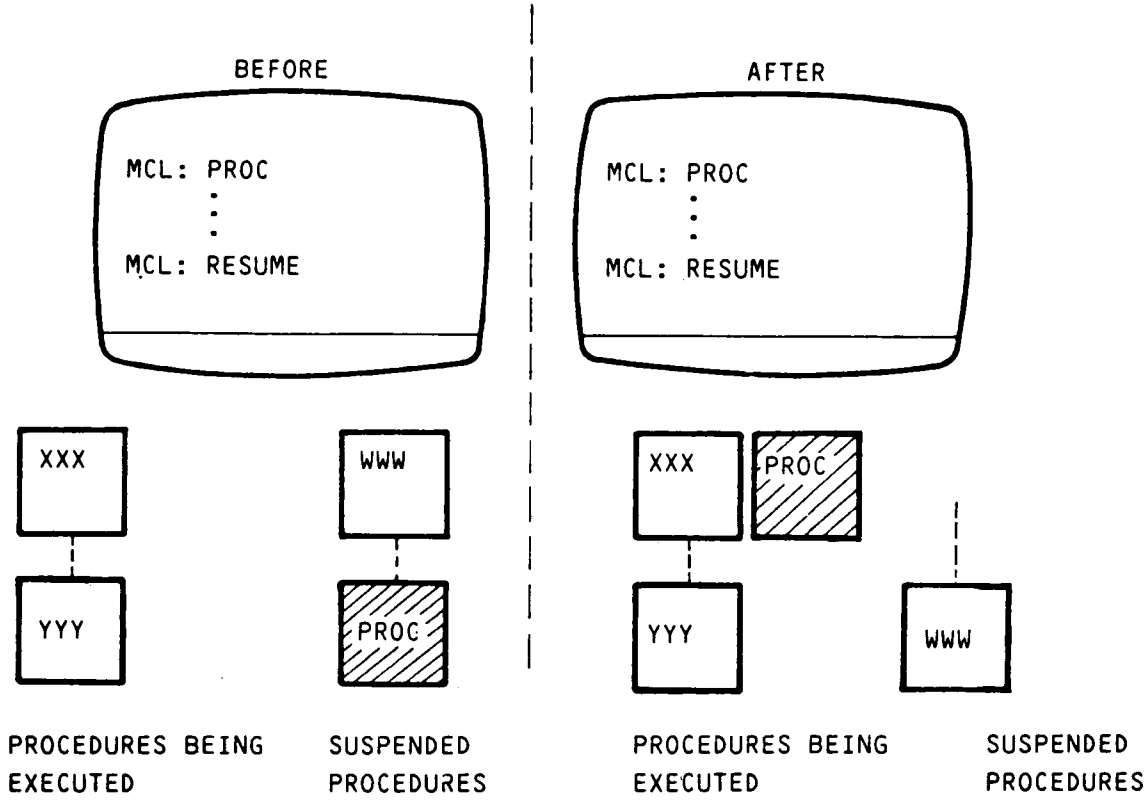
This command must be entered in the main window of the screen.

This command has no parameters.

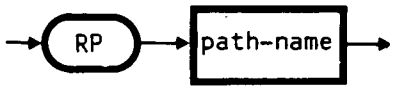
Note

See also the commands: KILL, SUSPEND.

Example



RP displays the physical file description from the Permanent Dataset Descriptor (PDD) of a specified file system object on disc.



where:

path-name identifies a specific file on a mounted volume.

Characteristics

1. RP may only be used non-interactively.

2. Access rights required to execute the command are execute and read on the volume and on all the file system objects comprised between the volume and the object specified in the path name.

DISPLAY

The information obtained is:

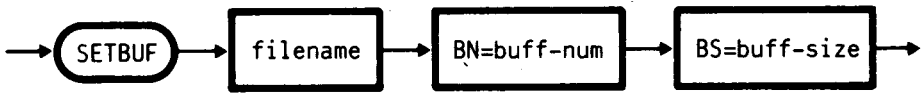
PDnn	Number of the specific Permanent Descriptor (PD) in the PDD TABLE.
DATASETID	Name of the PD in the PDD TABLE.
FILETP	Type of the file, which may be DEV, DIR, VOL, or BST.
ACCESS RIGHTS	Access rights to the specified file.
OWNERID	Owner identity.
LENGTH	File length in bytes.
ALLOC UNIT	Allocation unit of the specified file.
FIRST BYTE	Offset of the first byte of the file.
NEXT AVAILABLE PDD IN FREE LIST	If the PDD number is not zero, this is the first PDD in the free list. If it is zero, see below.
EXTENSION PTR	Not currently used.
DEVICE UNIT	Device number.
LINK COUNT	Number of files linked to the file specified.
CREATE TIME	Date of creation of the file.
LAST ACCESS TIME	Date of last access of the file.
LAST MODIFY TIME	Date of last update of the file.
NEXT AV EXTENT	Used if the PDD number is zero.
USER TYPE	Information given by the user and inserted when the object is created.
RESERVED	Reserved for future use.

Assigns private buffers for I/O operations on global files, which are usually executed by means of a system buffer pool.

SETBUF is to be used only once for each file and cannot be used to pass from one private buffer pool to another.

SETBUF cannot be used if I/O operations are in progress.

Buffer assignment stops with the last disconnection from the file (see DISCON command, in MCL MOS Command Language User Guide).



where:

filename identifies the file to which the private buffers are allocated. If it does not begin with the character "/", the file identifier is first searched for in the context table (see MOS Programmer Guide) and then in the working directory.

buff-num specifies the number of buffers required for the private pool buffer.

buff-size specifies the size in bytes of each private buffer.

The command returns a completion code in the variable "%STATUS", whose possible values are:

VALUE	MEANING
0	Correct operation.
1	Unable to reserve private buffers.
2	Non-existing file.

Characteristics

1. The value of the two keyed parameters must be compatible with the configuration definition.

2. The file identifier may be a local name if a connect operation to the corresponding global name has been made previously.

3. The private buffers assignment is done in the same way as by the "CONN" command (see MCL MOS Command Language User Guide). The only difference is that a completion code is returned.

4. Access right for execution on the father directory is required to use this command.

Allows the user to modify the attributes of one or more files.

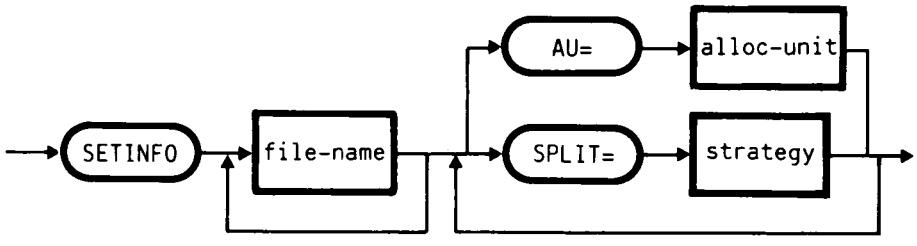
If the list of files specified in the command contains files of different types, the new attributes specified must be compatible with all the files in the list.

The attributes which may be modified are:

- the allocation unit of the file
- the division strategy of the pages of the B*-tree in the primary index of a keyed file.

This command does not allow directory attributes to be modified.

The table on the following page specifies the attributes which may be modified for each file type.



where:

file-name is the name or path name of the file.

alloc-unit is the new allocation unit for the file which must be in the range $2^{31}-1$. The number specified is rounded up to the nearest multiple of 512.

strategy is the new division strategy for the pages of the B*-tree. Possible values are: 50, 90, 100.

TYPE/ATTRIBUTE	AU	SPLIT
Byte-stream file	X	-
Positional file	X	-
Keyed file	X	X

Tab. 1 Attributes Modifiable with SETINFO

Characteristics

1. If the allocation unit parameter is set to 0, no further expansion of the file will be possible, that is, data may not be appended to the file.

2. All errors that may occur during the execution of SETINFO are displayed as follows:

   SETTING file-name INFO: error message

   where "error message" indicates the nature of the error (see Appendix A).

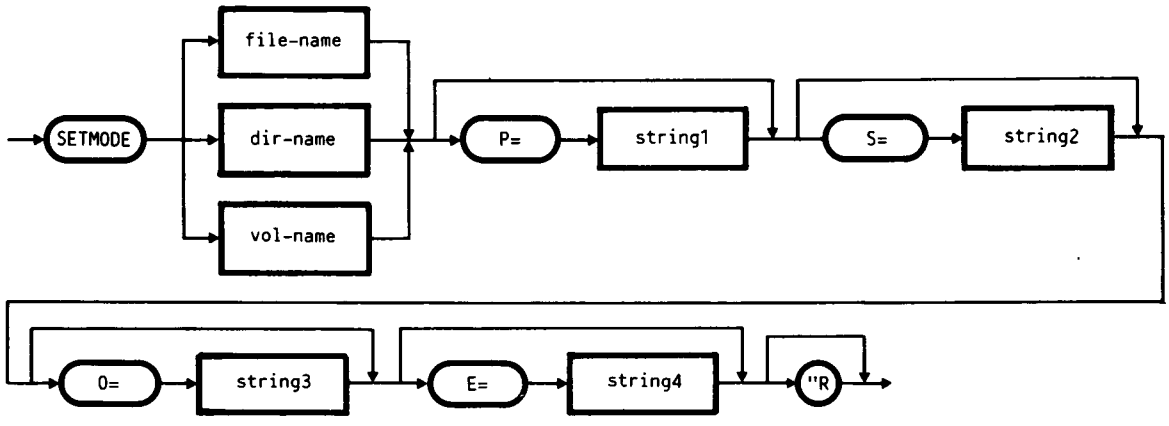
3. The following access rights are required to perform SETINFO:

   - execute on the father directory
   - write on the file.

This command allows access rights of the following file system items to be modified:

- files
- directories
- volumes.

The access rights to a file system item can only be changed by the owner of the item or the system administrator (ROOT).



where:

file-name is the name or path name of the file.

dir-name is the name or path name of the directory.

vol-name is the name of path name of the volume.

string1 is the string specifying the access rights to assign to the owner (see first note).

string2 is the string specifying the access rights to assign to the group users (see first note).

string3 is the string specifying the access rights of all the other users (see first note).

string4 represents the execution bits for primary and secondary protection (see second note).

"R is an option which enables modification of the access rights and execution bits for all the objects contained in the a directory, in recursive fashion.

Note

1. The values to be given in "string1", "string2", and "string3" for the input parameters **P**, **S**, and **O** respectively are as follows:

   . any combination of the letters **R** (read), **W** (write), **A** (append), and **X** (execution)

   . the character _ (underline), which cancels all access rights.

2. The value to specify in "string4" for the parameter **E** can be as follows:

   - **P** to assign the primary protection execution bit

   - **S** to assign the secondary protection execution bit

   - **PS** to assign the primary and secondary protection execution bits

   - the character _ (underline), to remove the existing execution bits.

   If the execution bit input parameter **E** is not given, the execution bits are not modified.

Characteristics

1. If the file name identifies a positional or keyed file, the command SETMODE modifies the access rights of all the associated files. For example, the same access rights are assigned to a keyed file and to the corresponding index files.

2. The recursive option **"R** is only significant if the SETMODE command is performed on a directory.

3. When a file system object is created, it is assigned the access rights defined in the list of default values.

Examples

1. The following command:

   SETMODE FILE P=RX S=X O=_

   assigns the following access rights to "FILE":

   - read and execute for the owner
   - execute for the group users
   - none for all the other users.

2. The following command:

   SETMODE DIR S=RX O=_

   modifies the access rights to the directory "DIR" as follows:

   - the owner's access rights do not change
   - group users have read and execute access rights
   - all the other users have no access rights.

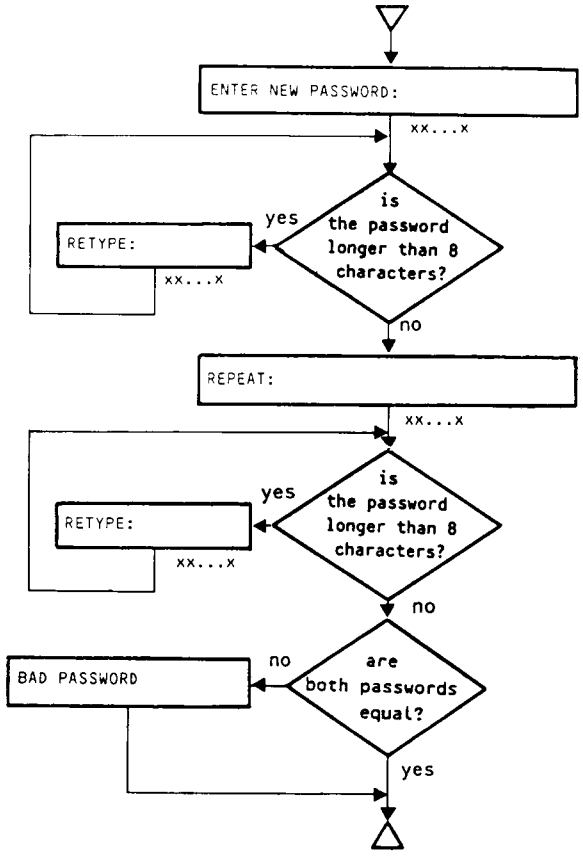
Allows the user to define or change his password consisting of up to 8 alphanumeric characters (A..Z, a..z, 0..9, ., and _).

The command has two different modes of operation.

When entered, the password is not displayed on the screen.

The system requests the parameters interactively (see operator interface).

Defining a Non-Existent Password



ENTER NEW PASSWORD:

Requests the new password to be entered.

xx...x Represents the new password.

RETYPE:

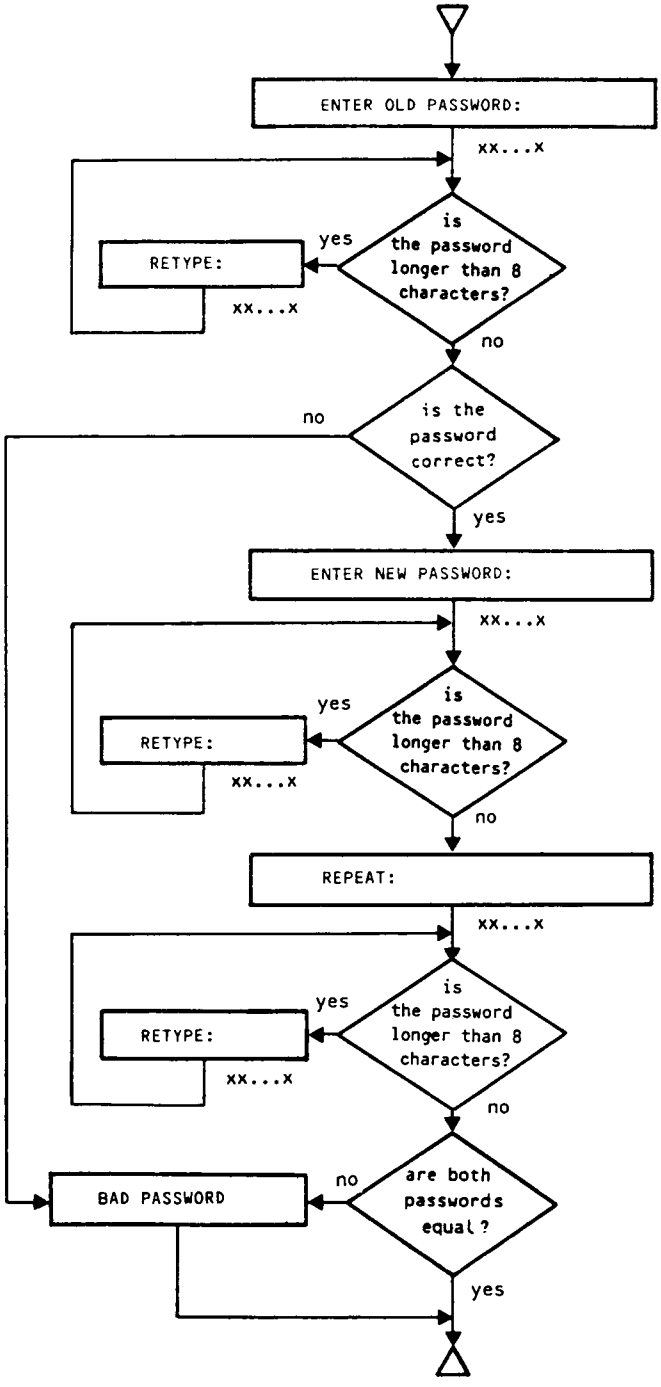
A nine character password was entered by mistake. The password cannot have more than 8 characters. Re-enter the password.

xx...x Represents the correct password.

REPEAT:

Requests the new password to be re-entered for confirmation.

xx...x Represents the new password, retyped for confirmation.


ENTER OLD PASSWORD:
xx...x
is the password longer than 8 characters?
yes
RETYPE:
xx...x
no
is the password correct?
no
yes
ENTER NEW PASSWORD:
xx...x
is the password longer than 8 characters?
yes
RETYPE:
xx...x
no
REPEAT:
xx...x
is the password longer than 8 characters?
yes
RETYPE:
xx...x
no
are both passwords equal?
no
BAD PASSWORD
yes

ENTER OLD PASSWORD:

Requests the entry of the old password to ensure that the user is authorised to change the password.

xx...x Represents the old password.

ENTER NEW PASSWORD:

Requests the new password to be entered.

xx...x Represents the new password.

REPEAT:

Requests the new password to be re-entered for confirmation.

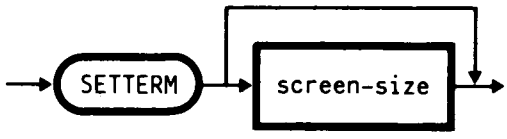
xx...x Represents the new password, re-typed for confirmation.

RETYPE:

A nine character password was entered by mistake. The password cannot have more than 8 characters. Re-enter the password.

xx...x Represents the correct password.

Sets the physical screen size of the current terminal.



where:

screen-size specifies the new screen size of the terminal, which may be 520, 1000 or 2000 characters.

If no parameter is specified, the command requests the new screen size interactively.

The screen size parameter defines the number of rows and columns on the screen as follows:

PARAMETER	ROWS	COLUMNS
520	13	40
1000	25	40
2000	25	80

When SETTERM is activated, the screen is first cleared, then set to the new size.

Information Messages

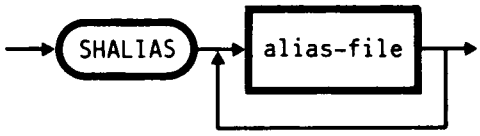
ENTER SCREEN_SIZE (520,1000,2000):

Enter the required screen size.

Note

The characteristics of an M24 terminal or a graphics screen cannot be modified with the SETTERM command.

Displays the contents of one or more alias files.



where:

alias-file is the name of the alias file to be displayed.

Characteristics

1. The complete path name of the aliased file is shown.
2. The following access rights are required to execute the command:
   - execute on the father directory
   - read on the alias file.

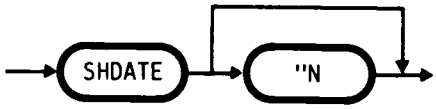
Example

The alias files output1 and output2 contain the path names of aliased files as follows:

```
MCL: SHALIAS output1 output2
     output1:
     /IPL/USR/ROOT/OUTPUT/FILE1
     output2:
     /IPL/USR/GENEV/RECORDS
MCL:
```

Displays the system date and time.

This information is also stored in the predefined variable %RET.



where:

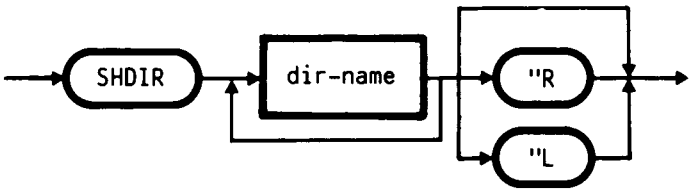
"N specifies that the date is not to be displayed on the screen, but only returned in %RET.

Example

```
MCL: SHDATE

     THU DEC 17   11:41:13  1987

MCL:
```

Lists the contents of one or more directories. By specifying the appropriate option the sub-directories or their contents may also be shown: a sub-directory will be identified by the character "(" after its name, or its whole contents may be displayed in parentheses.

If no directory name is specified, the current working directory contents are shown.



where:

dir-name is the name or path name of the directory.

"R is the option requesting display of the contents of all the sub-directories.

"L requests display of the sub-directory names only.

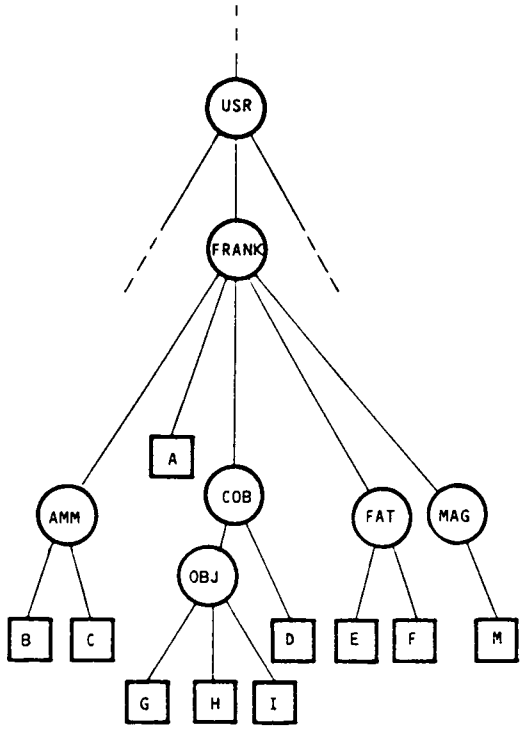
Characteristics

1. To execute this command, access rights to execute and read are required on the specified directory.

2. If the option **"R** is specified, access rights to read and execute are also required on all the sub-directories.

Examples

Supposing we have the following memory structure:

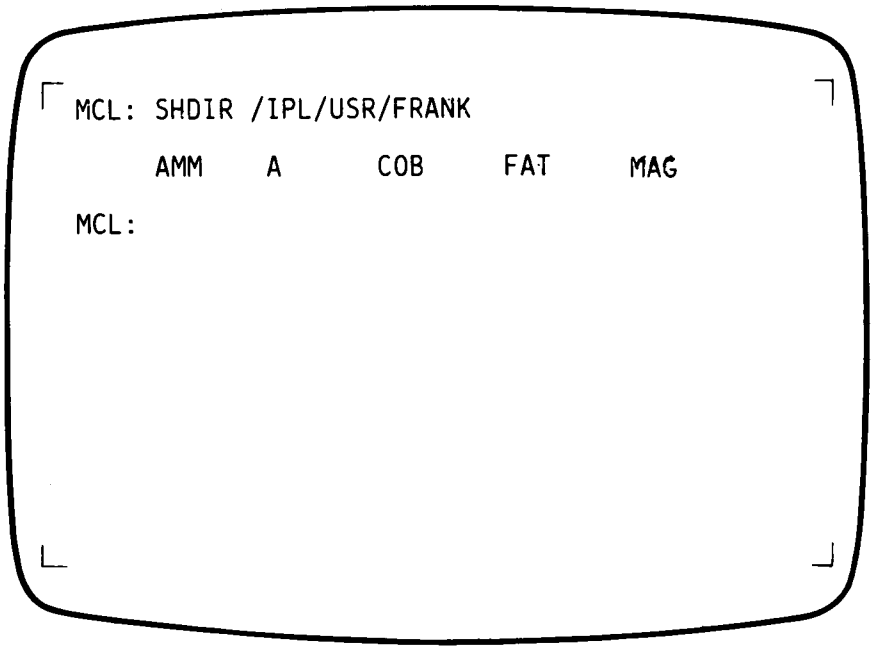


---

1. If the following command is entered:

SHDIR /IPL/USR/FRANK

the resulting display is as follows:



2. If the following command is entered:

```
SHDIR /IPL/USR/FRANK "R
```

the resulting display is:

```
MCL: SHDIR /IPL/USR/FRANK "R

     AMM(   B       C)     A
     COB(   OBJ(    D)     FAT(
     E      F)      MAG(   M)

MCL:
```

3. If the following command is executed:

```
SHDIR /IPL/USR/FRANK "L
```

the system displays:

```
MCL: SHDIR /IPL/USR/FRANK "L

     AMM(   A       COB(   FAT(
     MAG(

MCL:
```

Displays status information for all the process families in the system.

For each family, the following is displayed:

- the family number identifying the family
- the family number identifying the father family
- the priority number giving the priority of the family
- some general family information
- the login name of the user who started the family
- the name of the program loaded in the family's address space.

Background activities are taken into account by SHFAM.

This command has no parameters.

Characteristics

1. The contents of the "information" field vary according to whether the activity was started interactively or not:

   - interactive execution:

     . workstation name (TTYX, or VTX1 to VTX4, where X = A,B,... or 1..5) associated to the family

   - non-interactive execution:

     . output medium (usually the name of a disc file)

     . "BKGn" where "n" is in the range 1 to 9 (for programs activated directly by Grandpa).

2. If the following is displayed in the program name field:

   NOT AVAILABLE

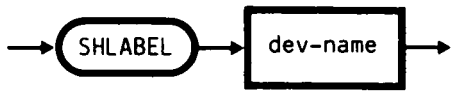
   a disconnect operation has been executed after loading and consequently the name of the program that has been loaded in memory cannot be retrieved at run-time.

Example

```
MCL: SHFAM

FAMILY     FATHER   PRIORITY    INFO     LOGIN    PROGRAMS
------     ------   --------   -------   -----    --------
   1          0          0                        NOT AVAILABLE
   2          1       2000      TTYB     ROOT     $LOG
   3          1       2000      TTYA     ROOT     $VSH
   4          3       2000      TTYA     ROOT     SHFAM
   5          1       2000      VTC1     USR1     LIST

MCL:
```

Displays the descriptor contents of the main volume of a hard or floppy disc located (but not mounted with the MNT command) on the specified device.

The volume name is stored in the predefined variable %RET.



where:

dev-name is the name of the device on which the volume is inserted. Valid devices are: FL1 to FL4, HD1 to HD8, MF1 to MF4 (for 1 Mbyte mini-floppies), and MF5 to MF8 (for 320 Kbyte mini-floppies).

Characteristics

Read access right on the specified device is required to execute this command.

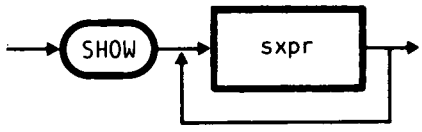
Example

```
MCL: SHLABEL FL1

VOLUME DESCRIPTOR

VOLUME LABEL      : VOL
RELEASE NUMBER    : 1
RELEASE LEVEL     : 2
VOLUME BLOCK SIZE: 512
VOLUME SIZE       : 984576
NUMBER OF FILE    : 100
```

Displays the strings resulting from the calculation of simple expressions (possibly keyed parameters or other variables) specified in the command line.

Only the value of variables is shown on the screen at execution time.

The command can be redirected.



---

where:

sxpr is a simple expression, as described in the MCL MOS Command Language User Guide.

Characteristics

1. When used within an MCL procedure, this command allows the user to follow the procedure execution. By inserting the SHOW command at various stages of the procedure, the sequence of messages may be followed on the screen.

2. In interactive mode, the machine may be used in the same way as a desk-top calculator: having entered the expression to be calculated, the result is displayed on the screen on the line following the command line.

3. The strings resulting from the simple expressions may be redirected to a file, another terminal (physical or virtual), or the workstation printer.

4. The strings making up the expression to be displayed must be separated by a space; quotes are optional.

Note

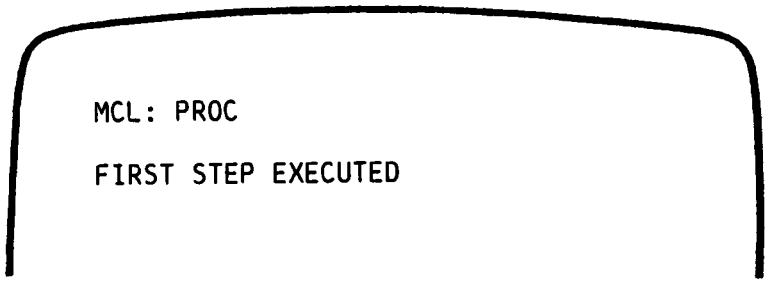
See also the commands: ECHO, WRITE.

Examples

1. The PROC procedure contains the command:

```
SHOW "FIRST STEP EXECUTED"
```

   After the procedure has been activated and the command executed, the following will appear on the screen:



---

2. If MESSAGES is a byte-stream file, after the execution of the command:

```
SHOW STEP 1 EXECUTED >> MESSAGES
```

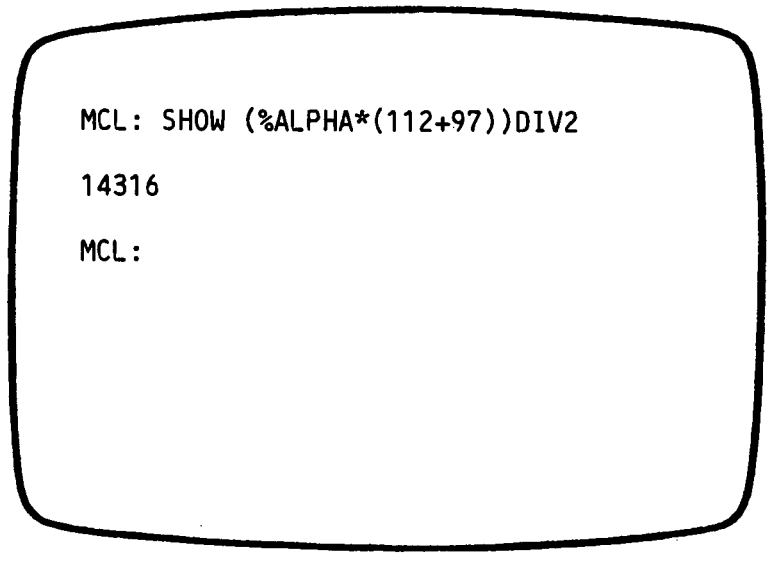
   the specified string is appended to the MESSAGES file.

3. If the following command is entered:

```
SHOW SESSION WILL CLOSE IN 5 MINS > /DEV/TTYB
```

   the specified message appears on the screen of the TTYB terminal.

4. If SHOW is used interactively with the variable %ALPHA having, for example, the value 137, the following will appear on the screen:



Displays the current characteristics of the terminal, physical as well as virtual.

This command has no parameters.

Characteristics

When activated, SHTERM displays the following information:

```
/DEV/TTYNAME

WS:
PHYSSIZE:      2000
CTRMODE:       HIDE NORMAL
SCREEN:        0
KEYS:          11101010
```

---

where:

/DEV/TTYNAME is the name of the terminal: TTYA, TTYB, ... for the physical terminal, or VTX1 to VTX4 for virtual terminals (where X = A, B,...).

PHYSSIZE is the number of characters that the video can display.

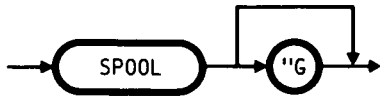
CRTMODE describes the visual attributes of the video (reverse, blink,...).

SCREEN is the video type (3 = alphanumeric, 0 = graphic)

KEYS represents the current setting of the key locks (if any) on the keyboard. The first two figures are not used, and the next three pairs indicate the setting of the three keys:

11 = vertical
01 = turned right
10 = turned left

SPOOL allows "normal" users some control over print jobs they have submitted to the spooling system.



where:

"G is an option enabling management of the global SPOOL system in a distributed configuration.

SPOOL FUNCTIONS

Normal users may perform the following operations:

- display the status of a class (L)
- display the status of a print job (J)
- set the status of a print job to HOLD (H)
- set the status of a print job to READY (R)
- remove (kill) a print job from a class (K)
- suspend the action of a specified printer (P)
- show the spooling system environment (S)
- return to the Shell environment (E).

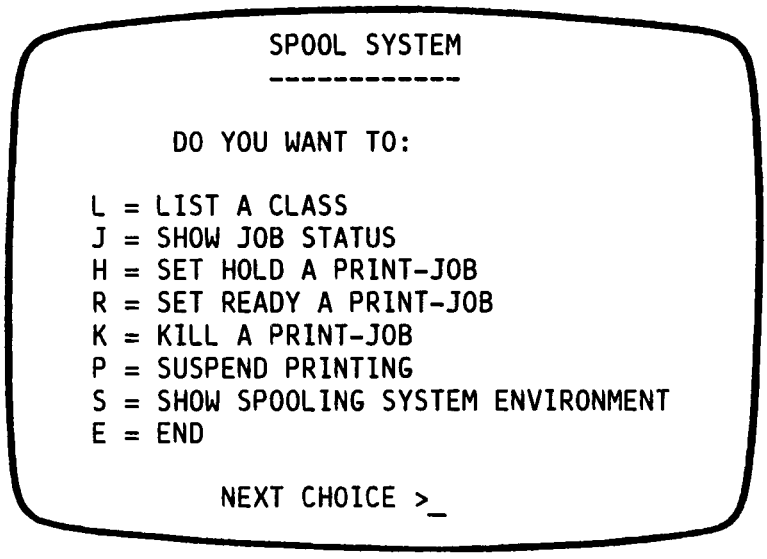


Fig. 3.SPOOL-1 SPOOL Command - Normal User Menu

Select the function by entering the appropriate letter. (**H, R, K,** and **P** are permitted only to the print job owner and the system administrator).

The menu is then replaced by a request for a parameter, (see table and details below).

After a function has been performed, the user may perform another.

The user may return to the main menu by typing **.CR** at any time during parameter input or at the end of a function

Alternatively, the user can trigger the function immediately by keying in the corresponding letter code, followed by the required parameters (separated by blanks).

Parameters

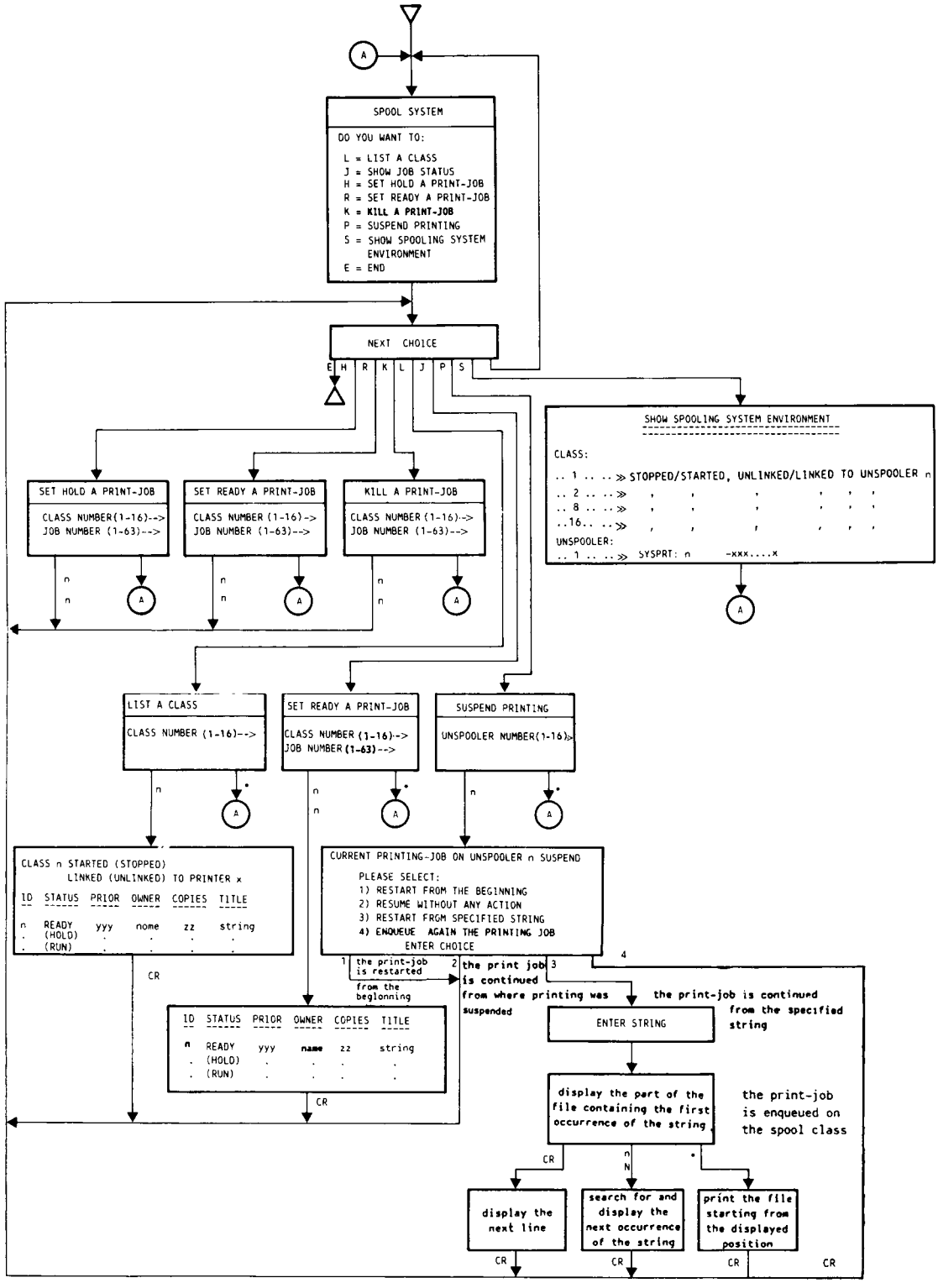
The parameters required by the SPOOL functions are shown in the following table:

SPOOL FUNCTION	COMMAND LETTER	PARAMETERS REQUIRED Parameter1	Parameter2
list a class	L	class	-
show job status	J	class	job number
set hold a print job	H	class	job number
set ready a print job	R	class	job number
kill a print job	K	class	job number
suspend printing	P	unspooler	-
show spooling system environment	S	-	-

Tab. 1 SPOOL Function Parameters

Valid parameter values are as follows:

- unspooler: a number from 1 to 16
- class: a number from 1 to 16
- job number: a number from 1 to 63, as allocated by SPOOL when the job is submitted.


A
SPOOL SYSTEM
DO YOU WANT TO:
L = LIST A CLASS
J = SHOW JOB STATUS
H = SET HOLD A PRINT-JOB
R = SET READY A PRINT-JOB
K = KILL A PRINT-JOB
P = SUSPEND PRINTING
S = SHOW SPOOLING SYSTEM ENVIRONMENT
E = END
NEXT CHOICE
E H R K L J P S
SET HOLD A PRINT-JOB
CLASS NUMBER(1-16)-->
JOB NUMBER (1-63)-->
SET READY A PRINT-JOB
CLASS NUMBER (1-16)->
JOB NUMBER (1-63)-->
KILL A PRINT-JOB
CLASS NUMBER (1-16)-->
JOB NUMBER (1-63)-->
SHOW SPOOLING SYSTEM ENVIRONMENT
CLASS:
.. 1 .. .. >> STOPPED/STARTED, UNLINKED/LINKED TO UNSPOOLER n
.. 2 .. .. >>
.. 8 .. .. >>
..16.. .. >>
UNSPOOLER:
.. 1 .. .. >> SYSPRT: n -xxx....x
LIST A CLASS
CLASS NUMBER (1-16)-->
SET READY A PRINT-JOB
CLASS NUMBER (1-16)->
JOB NUMBER (1-63)-->
SUSPEND PRINTING
UNSPOOLER NUMBER(1-16)>
CLASS n STARTED (STOPPED)
LINKED (UNLINKED) TO PRINTER x
ID STATUS PRIOR OWNER COPIES TITLE
n READY yyy nome zz string
. (HOLD)
. (RUN)
CR
CURRENT PRINTING-JOB ON UNSPOOLER n SUSPEND
PLEASE SELECT:
1) RESTART FROM THE BEGINNING
2) RESUME WITHOUT ANY ACTION
3) RESTART FROM SPECIFIED STRING
4) ENQUEUE AGAIN THE PRINTING JOB
ENTER CHOICE
1 the print-job is restarted from the beglonning
2 the print job is continued from where printing was suspended
3 ENTER STRING
4 the print-job is continued from the specified string
ID STATUS PRIOR OWNER COPIES TITLE
n READY yyy name zz string
. (HOLD)
. (RUN)
CR
display the part of the file containing the first occurrence of the string
the print-job is enqueued on the spool class
CR
n N
*
display the next line
search for and display the next occurrence of the string
print the file starting from the displayed position
CR
CR
CR
CR

where:

ID is the print job identification number allocated by the spooler.

STATUS is the current status of the print job (either HOLD, READY or RUN).

PRIORITY is the priority of the print job. The job with lowest priority number is printed first.

OWNER is the name of the user submitting the print job.

COPIES is the number of copies of the print job requested.

TITLE is the character string given as print job title by the user submitting the job.

NEXT CHOICE --->

X [n]	Selects another SPOOL function (with or without appended parameter).
.	Returns to the main menu.
E	Exits from SPOOL and returns to Shell.

CLASS NUMBER (1-16)--->

n	Represents the number of the required spool class in the range 1-16.
.	Returns to the main menu.

JOB NUMBER (1-63)--->

nn	Represents the required print job number.
.	Returns to the main menu.

UNSPOOLER NUMBER (1-16)--->

n	Represents the required unspooler number in the range 1-16.
.	Returns to the main menu.

CURRENT PRINTING JOB ON UNSPOOLER n SUSPENDED

PLEASE SELECT:

1) RESTART FROM THE BEGINNING
2) RESUME WITHOUT ANY ACTION
3) RESTART FROM A SPECIFIED STRING
4) ENQUEUE AGAIN THE PRINTING JOB

ENTER CHOICE >

The unspooler with number "n" has been suspended during printing, and the user is allowed to indicate at what point he wants the print restarted by selecting one of the following:

1	to restart at the beginning of the print job
2	to restart the print job where it was interrupted
3	to restart from a specified string
4	to enqueue again the print job just interrupted.

ENTER STRING:

This is output when option **3** above has been selected, to prompt for the string from which printing must resume. SPOOL searches for the string and then displays the screen page in which it appears for the first time. Now:

CR	displays another page
N or n	searches for the next occurrence of the string (if there are none, nothing is done)
.	starts printing again from the first character displayed at the top left, and continues to the end of the file.

Note

When printing is interrupted, requests to put the job back in the spooling queue do not affect its attributes (priority, status, title, etc.)

Suspends the job in execution.

The system returns to Shell and any command, program, or procedure may now be specified and executed. To resume the execution of the suspended job, use the RESUME command.

This command must be entered on the system line.

This command has no parameters.

Characteristics

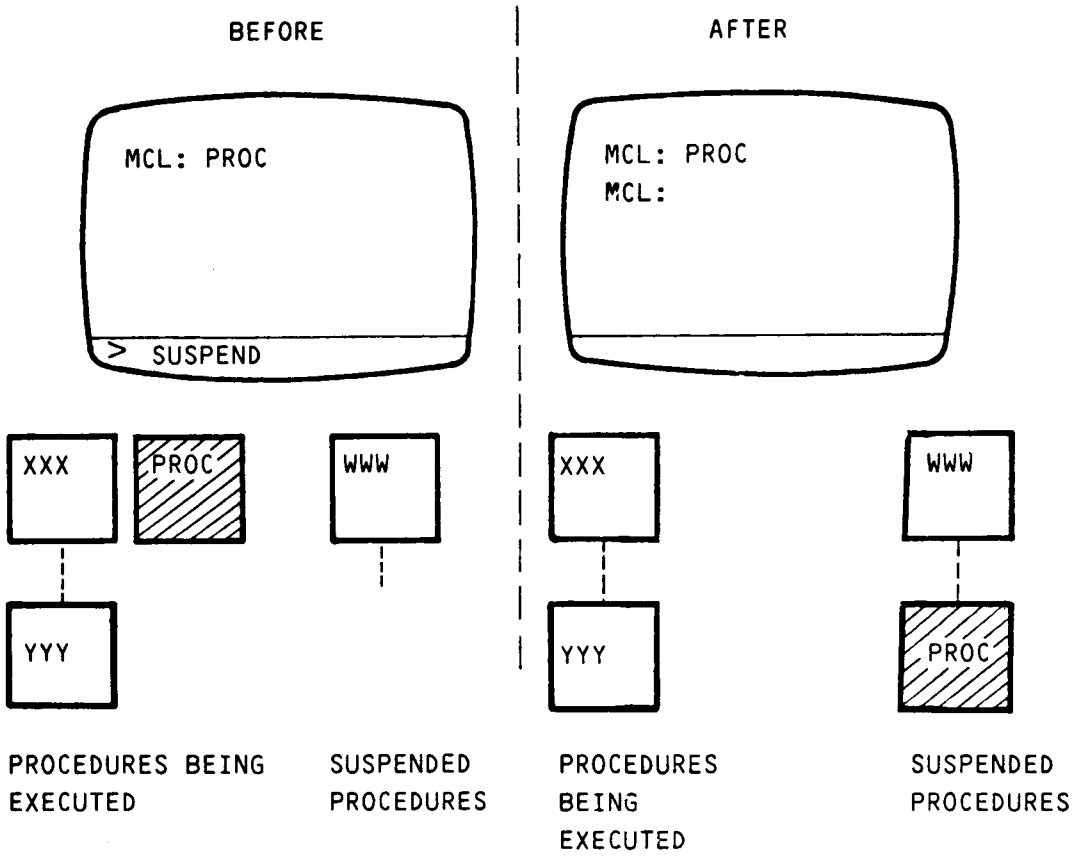
A maximum of four jobs can be suspended. If the user attempts to suspend a fifth job, the following error message appears:

MCL ERROR 38: INVALID OPERATION

Notes

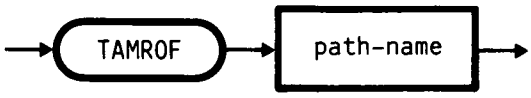
1. See also commands: KILL, RESUME.
2. Suspended files are not available to other active processes.

Example


BEFORE
AFTER
MCL: PROC
> SUSPEND
MCL: PROC
MCL:
XXX
PROC
WWW
YYY
XXX
YYY
WWW
PROC
PROCEDURES BEING EXECUTED
SUSPENDED PROCEDURES
PROCEDURES BEING EXECUTED
SUSPENDED PROCEDURES

Enables a keyed file or a positional file with record deletion permitted to be converted from 6.0 format to 5.2 format, so that it becomes compatible with the file system of release 5.2.

This command is issued on a machine at release 5.2, and is the converse of the command FORMAT.



---

where:

path-name is the local path name of the file to be converted; it cannot be the name of an alias file.

The Tape Conversion Utility (TCU) allows exchange of data between the L1 and other systems which use magnetic tapes recorded according to ECMA 13 (Olivetti 32), IBM, or UNLABEL standards.

Magnetic tapes recorded on other systems can be converted to L1 system format and then processed. Files processed by an L1 system can be converted and stored on tapes that can be read by other systems.

The functions provided by TCU are as follows:

- READTAPE: to read data files recorded on magnetic tape
- WRITETAPE: to record data files on magnetic tape.

Characteristics and Restrictions

1. The TCU supports magnetic tapes recorded at 1600 bits/in density.
2. One (UNLABEL, ECMA or IBM) or more (ECMA or IBM) files can be transferred at the same time. Files can either be recorded at the beginning of the tape or appended after the last file.
3. Files with spanned records cannot be recorded: spanned records are records which do not finish at the end of the block but continue in the next one.
4. Any control characters at the beginning of blocks are ignored.
5. The record length must be a factor of the length of the block defined for the file.
6. The block length is its actual length, not the maximum number of characters it can contain.
7. Only numbers, upper case letters, blanks, and the following special characters:

   ! " % & ' ( ) * + , - : ; < = > ? _

   can be specified in the file labels.

Note: For a description of magnetic tapes, see Appendix C.

File Structure and Type

When the UNLABEL standard is used, only byte-stream files can be converted to and from magnetic tapes. These contain unpacked characters and may be either data files or object files. When conversion between ASCII and EBCDIC codes is required, packed data files are also allowed.

L1 MOS supports the ASCII code. The EBCDIC code can be converted into ASCII by user request.

A file can be continued onto several tapes; in this case, the user is responsible for leading the tapes in the correct sequential order.

Trace File

The user may specify a trace file in which to store the details of the conversion: this is particularly useful when several conversions are carried out. The trace file can be listed or printed using the normal Shell commands.

The following information is recorded in the trace file each time a conversion is carried out:

- the standard
- the function requested
- the parameters
- the result.

The information is stored in the trace file in the following way:

```
TAPE CONVERSION UTILITY PARAMETERS :
     STANDARD TYPE                 = 2
     FUNCTION                      = W
     VOLUME NAME                   = REGISTRY
     OWNER NAME                    = WORSLEY
     WRITE MODE                    = C
     INPUT FILE NAME               = LONDON
     OUTPUT FILE NAME              = LONDON22
     BLOCK LENGTH                  = 00254
     DATA FORMAT                   = A
***  RESULT = CORRECT TERMINATION  ***
```

OPERATION

The parameters are requested interactively.

The default values are selected by pressing **CR** in reply to a prompt.

The interactive requests depend on the function required (READTAPE or WRITETAPE) and on the recording standard used on the magnetic tape.

READ TAPE FUNCTION

This function reads files from a magnetic tape onto an L1 system.

UNLABEL: The information requested is as follows:

- the number of the drive on which the tape is mounted
- the order number in which the file has been recorded
- the path name of the output file for the data read and converted
- the data block size
- the data format (ASCII or EBCDIC).

ECMA/IBM: The information requested is as follows:

- the number of the drive on which the tape is mounted
- the name of the tape volume
- the name of the tape owner
- the name of the file to read
- the path name of the output file for the data read and converted
- the data format (ASCII or EBCDIC), only for ECMA standard.

WRITE TAPE FUNCTION

This function records files on a magnetic tape in an L1 system.

File recording can either start at the beginning of the tape, thus cancelling any existing files, or be appended to the end of the last file.

UNLABEL: The information requested is as follows:

- the number of the drive on which tape is mounted
- the recording mode (start of file or appending)
- the name of the file to convert and record on tape
- the data block size
- the data format (ASCII or EBCDIC).

ECMA/IBM: The information requested is as follows:

- the number of the drive on which the tape is mounted
- the name of the tape volume
- the name of the tape owner
- the recording mode (start of file or appending)
- the path name of the file to be recorded
- the name of the output file for the converted data
- the block length
- the record length
- the file creation date
- the file expiry date
- the data format (ASCII or EBCDIC).

Note

A prompt for the path name of an optional trace file, in which to store information on the conversion process both for READTAPE and WRITETAPE, also appears on the screen.

Conversion

When the READTAPE function is selected:

- the file required by the user is selected from those on the tape
- the parameters input are checked with the tape values; all errors are signalled on the screen
- if no errors are detected, the file data is read from the tape and stored in the L1 output file specified
- the details of the conversion are added to the trace file (if specified) and a message follows to indicate if the function has terminated correctly
- the user is asked if he wants to read another file and if so, a new set of requests is displayed in interactive mode.

When the WRITETAPE function is selected, conversion is done as follows:

- the L1 input file data is written on the magnetic tape
- the details of the conversion are added to the trace file (if specified) and a message follows to indicate if the function has terminated correctly
- the user is asked if he wants to record another file and if so, a new set of requests is displayed in interactive mode.

If an error occurs during the operation, the user can either repeat or abort the function.

Conversion Table

The conversion table for the READTAPE and WRITETAPE functions follows.

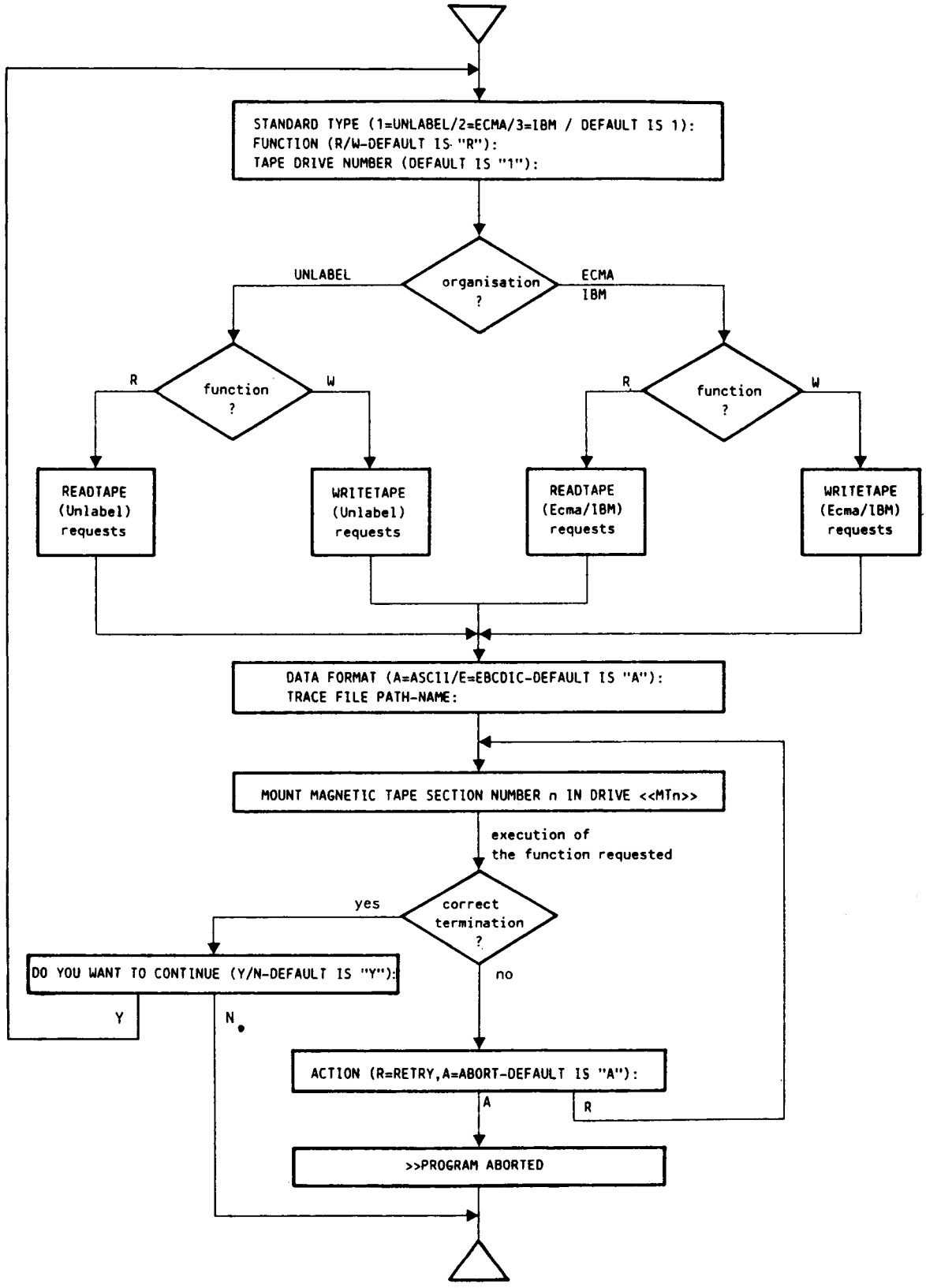
The user's reply to the question:

DATA FORMAT (A=ASCII/E=EBCDIC DEFAULT IS "A"):

can be either:

- **A** or just **CR** if conversion is not requested
- **E** if conversion is requested.

READTAPE				WRITETAPE			
Option A		Option E		Option A		Option E	
code on tape	code on L1 MOS	code on tape	code on L1 MOS	code on L1 MOS	code on tape	code on L1 MOS	code on tape
ASCII	ASCII	ASCII	EBCDIC	ASCII	ASCII	ASCII	EBCDIC
EBCDIC	EBCDIC	EBCDIC	ASCII	EBCDIC	EBCDIC	EBCDIC	ASCII


STANDARD TYPE (1=UNLABEL/2=ECMA/3=IBM / DEFAULT IS 1):
FUNCTION (R/W-DEFAULT IS "R"):
TAPE DRIVE NUMBER (DEFAULT IS "1"):
organisation ?
UNLABEL
ECMA
IBM
function ?
R
W
READTAPE (Unlabel) requests
WRITETAPE (Unlabel) requests
READTAPE (Ecma/IBM) requests
WRITETAPE (Ecma/IBM) requests
DATA FORMAT (A=ASCII/E=EBCDIC-DEFAULT IS "A"):
TRACE FILE PATH-NAME:
MOUNT MAGNETIC TAPE SECTION NUMBER n IN DRIVE <<MTn>>
execution of the function requested
correct termination ?
yes
no
DO YOU WANT TO CONTINUE (Y/N-DEFAULT IS "Y"):
Y
N
ACTION (R=RETRY,A=ABORT-DEFAULT IS "A"):
A
R
>>PROGRAM ABORTED

STANDARD TYPE (1=UNLABEL / 2=ECMA / 3=IBM / DEFAULT IS 1):

Requests the tape organisation:

1	UNLABEL
2	ECMA
3	IBM.

FUNCTION (R/W - DEFAULT IS "R"):

Select the function READTAPE or WRITETAPE:

R	READTAPE function
W	WRITETAPE function.

TAPE DRIVE NUMBER (DEFAULT IS "1"):

Requests the magnetic tape drive identifier.

n	Selects drive "n".
CR	Selects the default drive "1".

TAPE FILE NUMBER (DEFAULT IS "001"):

Enter the order number that identifies the file to be read on the magnetic tape. The number is not absolute, but relates to the last file read.

nnn Identifies the nnn-th file on the tape.

CR Identifies the first file, by default.

OUTPUT PATH-NAME:

Enter the path name of the file that is to receive the converted data. If the file does not exist, it is created. If it exists, it is overwritten.

xx...x Represents the file path name.

BLOCK SIZE (DEFAULT IS "512"):

Specify the block size. TCU rounds up the length of the output file to a multiple of the block size.

nn...n Represents the block size; it must be an even number from 18 to 16384 bytes.

CR Selects the default value 512 bytes.

VOLUME NAME:

Enter the volume name, with a maximum of 6 characters.

xx...x Represents the volume identifier.

OWNER NAME:

Enter the name of the owner of the magnetic tape, with a maximum of 14 characters.

xx...x Represents the owner identifier.

INPUT FILE NAME:

Enter the name of the file that is to be read from tape, with a maximum of 17 characters.

xx...x Represents the file name.

OUTPUT PATH-NAME:

Enter the path name of the file that is to receive the converted data. If the file does not exist, it is created. If it exists, it is overwritten.

xx...x Represents the file path name.

WRITE MODE (A=APPEND / C=CLEAR TAPE AND WRITE) (DEFAULT IS "A"):

Specify the write mode.

A Append mode.

C Writes from start of file after erasing the tape.

INPUT PATH-NAME:

Enter the path name of the file to be converted or copied onto magnetic tape.

xx...x Represents the file path name.

BLOCK SIZE (DEFAULT IS "512"):

Specify the block size. TCU rounds up the length of the output file to a multiple of the block size.

nn...n Represents the block size; this must be an even number from 18 to 16384 bytes.

CR Selects the default value of 512 bytes.

VOLUME NAME:

Enter the volume name, with a maximum of 6 characters.

xx...x Represents the volume identifier.

OWNER NAME:

Enter the tape owner name, with a maximum of 14 characters.

xx...x Represents the owner identifier.

WRITE MODE (A=APPEND / C=CLEAR TAPE AND WRITE) (DEFAULT IS "A"):

Specify the write mode.

A Append mode.

C Writes from start of file after erasing the tape.

INPUT PATH-NAME:

Enter the path name of the file to be converted and copied onto magnetic tape.

xx...x Represents the file path name.

OUTPUT FILE NAME:

Enter the name of the file to receive the converted data.

xx...x Represents the name of the output file, with a maximum of 17 characters.

BLOCK SIZE (DEFAULT IS "512"):

Specify the block size. TCU rounds up the length of the output file to a multiple of the block size.

mm...m Represents the block size; this must be an even number from 18 to 16384 bytes.

CR Selects the default value 512 bytes.

RECORD SIZE (DEFAULT IS "mm...m"):

Enter the length of the record in bytes, which must be equal to or a factor of the block size.

nn...n is the record length

CR selects the default value, that is the block size.

CREATION DATE (YYDDD):

Enter 5 numeric characters to represent the creation date of the tape file.

yyddd The date format is:

"yy" = year
"ddd" = day (sequential order from 1 to 366).

EXPIRATION DATE (YYDDD):

Enter 5 numeric characters for the expiry date of the tape file.

yyddd The format for the date must be:

"yy" = year
"ddd" = day (sequential order from 1 to 366).

DATA FORMAT (A=ASCII / E=EBCDIC - DEFAULT IS "A"):

Specify the type of conversion requested (see conversion table):

A The data is not converted.

E The data is converted: from ASCII to EBCDIC or vice versa depending on the initial format.

CR Selects the default "no conversion" option.

TRACE FILE PATH-NAME

A file can be specified to receive the details of the conversion operations. If this file exists already, the information is appended to it. If the file does not exist, it is created automatically.

xx...x Represents the file path name.

CR Ignores the trace file option.

MOUNT MAGNETIC TAPE SECTION NUMBER n IN DRIVE <<MTn>>

Mount magnetic tape with section number "n" in drive "MTn" and press **CR**.

CR Executes the selected function.

DO YOU WANT TO CONTINUE (Y/N - DEFAULT IS "Y"):

The user is asked if he wants to perform another READTAPE or WRITETAPE operation. The two operations cannot be alternated and the same standard and conversion type are assumed. The interactive requests are therefore limited to a subset as follows:

READTAPE (UNLABEL)	- TAPE FILE NUMBER (DEFAULT IS "001"): - OUTPUT PATH-NAME: - BLOCK SIZE (DEFAULT IS "512"):
READTAPE (ECMA/IBM)	- INPUT FILE NAME: - OUTPUT PATH-NAME:
WRITETAPE (UNLABEL)	- INPUT PATH-NAME: - BLOCK SIZE (DEFAULT IS "512"):
WRITETAPE (ECMA/IBM)	- INPUT PATH-NAME: - OUTPUT FILE NAME: - BLOCK SIZE (DEFAULT IS "512"): - RECORD SIZE (DEFAULT IS "512"): - CREATION DATE (YYDDD): - EXPIRATION DATE (YYDDD):

ACTION (R=RETRY, A=ABORT - DEFAULT IS "A"):

An operation did not terminate correctly (e.g. attempt to read an incompatible tape) or an error has occurred.

A The function (READTAPE or WRITETAPE) is aborted and control passes to Shell.

R The user is referred to the question displayed before the unfinished action.

Information and Error Messages

---

file name >> FILE NOT FOUND

The file specified in "file name" has not been found. In a READTAPE (ECMA/IBM) operation, a file is searched for from the current file.

>> BEGIN COPY FILE

Conversion has started.

>> DATE ERROR

The date keyed-in is incorrect. It must be made up of five digits; the first two specify the year, the next three the day of the year (from 1 to 366).

>> END COPY FILE

Conversion has finished.

>> FILE NOT FOUND

The file identified by a number has not been found. In a READTAPE (UNLABEL) operation, the number identifying the file relates to the position of the current file.

>> INPUT FILE IS EMPTY

The L1 MOS file to be converted is empty.

>> INPUT FILE IS NOT BYTE_STREAM

The L1 MOS file to be converted is not a byte-stream file.

>> IT IS NOT STANDARD TAPE

The tape is not UNLABEL, ECMA, or IBM standard; e.g. there is a TM (Tape Marker) at the beginning of the tape.

>> NO MORE SPACE ON THE TAPE

There is not enough space on the tape to start a WRITETAPE operation.

>> OWNER NAME NOT EQUAL TO ENTERED NAME

The owner name entered by the user in a READTAPE operation is not the same as that on the tape.

>> PROGRAM ABORTED

An error has occurred that has stopped conversion.

>> RECORD SIZE MUST BE EQUAL TO OR A FACTOR OF BLOCK SIZE

The record size must be equal to or a factor of the block size specified by the user.

>> THERE ARE ILLEGAL CHARACTERS IN INPUT STRING

There are illegal characters in the tape label (see "Characteristics and Restrictions", note 7).

>> VALUE ERROR

The character string input is not in the requested format, a non alphanumeric character has been included or is not in the defined range.

>> VALUE TOO LONG

The character string input is too long.

>> VALUE TOO SMALL

The character string input is too short.

>> VOL1 LABEL NOT FOUND

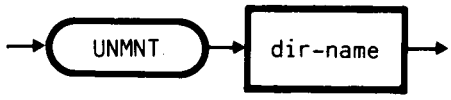
There is no "VOL1" label at the beginning of the tape.

>> VOLUME NAME NOT EQUAL TO ENTERED NAME

The volume name entered by the user in a READTAPE operation is not the same as that in "VOL1" on tape.

Cancels the logical connection between the physical medium mounted and the system, deleting the volume created for this purpose.

The medium (if removable) can be removed after the command has been executed.



where:

dir-name is the name of the directory which identifies the volume connection.

Characteristics

1. Before executing an UNMNT command the user must ensure that the working directory does not refer to the directory identifying the volume to be unmounted or one of its sub-directories.

2. To execute this command, access rights for reading and writing the directory that identifies the connected volume are required.

Note

See also the MNT command.

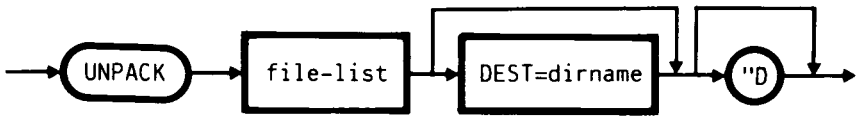
Examples

1. UNMNT /STORE

2. SET %VOL:=CLIENT
   SET %PTH:='/'
   UNMNT %PTH&%VOL

Unpacks one or more files that have been packed using the PACK command.

The suffix ".R" is removed from the file name.



where:

file-list is the list of files to unpack, each of which must have the suffix ".R".

dirname is the directory to contain the unpacked files; if this is omitted, the working directory is used instead.

"D is an option requesting removal of the packed files after unpacking.

Characteristics

1. If unpacking ends properly, the following message is displayed for each of the unpacked files:

   file-name: UNPACKED

2. The following message is displayed whilst the file is being unpacked:

   file-name: UNPACKING

3. The following access rights are required to execute UNPACK:

   - append and execute or write on the father directory to contain the unpacked output files.
   - read on the files to unpack.

USERMAN allows the "normal" user to list all the registered users and groups.

The parameters are requested in interactive mode.

USERMAN FUNCTIONS:

- List all users (in registration time order), whether they have defined a password or not, the groups to which they belong, and the machine on which they have been created.
- List all groups (in registration time order), their current user members and the machine on which they have been created.

Characteristics

1. More USERMAN functions are available to the system administrator "ROOT".
2. USERMAN works only in interactive mode.
3. The system administrator (ROOT) is the first and only user in a newly installed system.

OPERATION

```
        USER MANAGEMENT
        ---------------

DO YOU WANT TO:

        S = SHOW USERS
        L = SHOW GROUPS
        E = END
```

Fig. 3.USERMAN-1 USERMAN Command - Normal User Menu

To select a function the user must enter the appropriate letter and then press **CR.**

Listing functions S and L

These functions require no parameters, and their operation is as follows:

S produces the display of users shown below, which remains on screen until removed by a **CR.**

Users are listed in "historical" order, that is, the user registered earliest on the system is first on the list.

```
USER        PASSWORD      GROUP      INFORMATION     MACHINE NAME

User A          Y         Group 1    (information       NxMy
User B          Y         Group 4    on the users)      NxMy
User C          Y         Group 2                       NxMy
User D          N         Group 1                       NxMy

HIT <CR>
```

L lists groups and related information as shown below, also in "historical" order. The display remains on screen until removed by a **CR.**

```
GROUP       MACHINE NAME      MEMBERS

Group 1         NxMy          User A User C User M
Group 2         NxMy          User D User S
Group 3         NxMy          User V User K User J User X

HIT <CR>
```

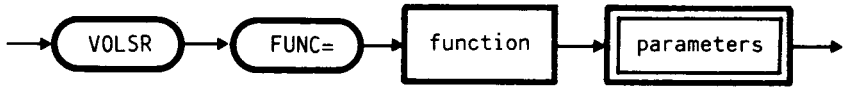
VOLSR dumps logical volumes from hard disc to Streaming Cartridge Tape (SCT4 or SCT5) or vice versa, without affecting the source volume.

VOLSR FUNCTIONS

SAVE	Saves a volume from a hard disc to SCT.
RESTORE	Restores a volume from SCT to hard disc.
VERIFY	Enables the user to check that the correct SCT is mounted, by displaying its identification details.
LABEL	Enables the user to initialise the SCT, by writing the first part of the tape header label (the "VOL1" section) at the beginning of the tape.
ERASE	Physically erases all labels and data from the SCT. Before further use, the tape will need to be initialised.

VOLSR operates both interactively and non-interactively.

Non-Interactive Mode

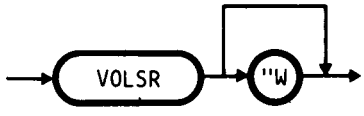


where:

function is one of the following: SAVE, RESTORE, VERIFY, LABEL, ERASE.

parameters is the set of parameters that must be specified for each function (see further on).

Interactive Mode



where:

"W is an option requesting the use of WRTALL to send a broadcast message to all connected users, asking them to end their operations, as VOLSR needs a dedicated system. When VOLSR finishes, another WRTALL broadcast informs users that they can resume work.

Characteristics

1. VOLSR requires a dedicated system (that is, it must have the entire system to itself, excluding all other operations) for its SAVE and RESTORE operations.

2. VOLSR permits more than one volume to be saved on a single SCT (multi-volume tape), or a large volume to be saved on more than one SCT, (single-volume tape). See Appendix C for the tape organisation.

3. VOLSR cannot handle physical or remote volumes.

4. A SAVE always starts writing on the first free block on the SCT, in append mode. Volumes with the same name as recorded previously can therefore be added to the same tape. Such volumes are then identified by an "occurrence" number (1 to 99). The first occurrence of tape volume always has an occurence number of 1.

5. When using RESTORE, the path name of the object to restore must be the same as that specified in the corresponding SAVE operation.

6. In the following text, use is made of the expressions "single-volume" and "multi-volume", referring to the mode of use of the tape:

   - A "single-volume" tape is a tape containing a dump of one single logical volume. Due to the size of this dump, more than one tape may be required to complete it.

   - A "multi-volume" tape describes a tape containing more than one logical volume or versions of the same volume. None of these volumes may be continued onto another tape.

7. The recording mode depends on the tape header label, and may be single or double (**S** or **D**).

   The double-recording mode writes the volume twice on the same tape (if this is not an SCT5), as a back-up measure (see below).

8. Double-recording mode uses twice as much tape. Thus, while the maximum file size that may be stored on a SCT4 in single-recording mode is approximately 18.8 Mbytes, the maximum file size that may be stored on a SCT4 in double-recording is only 9.4 Mbytes (see Appendix C).

9. The access rights assigned to the volume recovered on hard disc, are as follows:

   - if the volume is created, the access rights defined in the default access list are assigned

   - if the volume is overwritten, the access rights to the recovered volume are the same as for the existing volume.

10. The access rights and the owner of the objects contained in a volume are not modified following a volume restore operation.

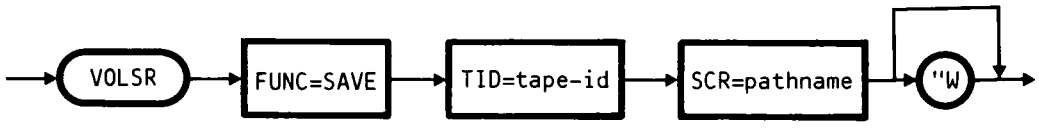
11. If a tape already labelled on a SCT4 drive is mounted on a "slim" (SCT5) drive for a save operation, the system will refuse the command with:

SAVE:PERMISSION DENIED.

NON-INTERACTIVE VOLSR

In non-interactive mode, VOLSR terminates a function before requesting any user interaction. The functions and parameters required are shown below.

SAVE FUNCTION



where:

tape-id is the identity of the tape required by the user (maximum of 6 characters), checked by the system to ensure that the correct tape is mounted.

pathname is the path name of the logical disc volume to save (must not be a remote volume, nor "/IPL"); multiple path names (maximum of 10) must be separated by blanks and the list enclosed in quotes.

"W is an option requesting the use of WRTALL to send a broadcast message to all connected users, asking them to end their operations, as VOLSR needs a dedicated system. When VOLSR finishes, another WRTALL broadcast informs users that they can resume work.

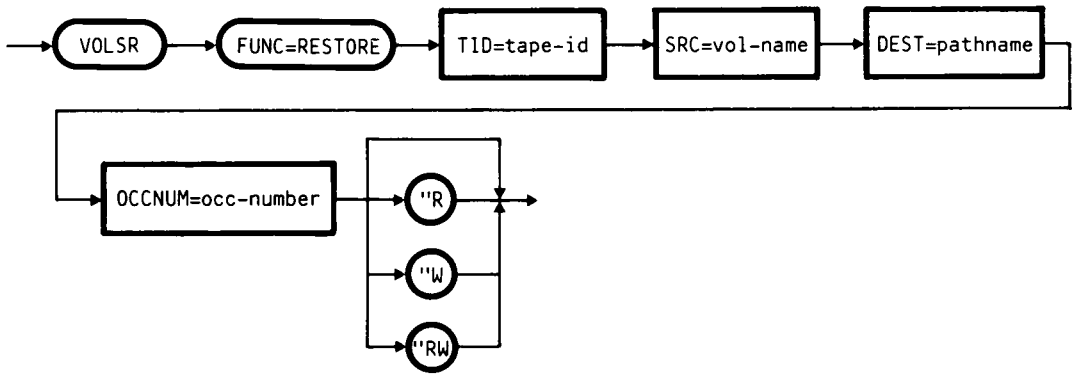
Characteristics

1. Whilst a volume is being saved, its name and size (in bytes and sectors) are displayed. In addition, in double-recording mode, the numbers of the tracks used for the back-up copy of the volume onto tape are displayed.

2. If saving a volume requires more than one tape, the following message appears when operator interaction is required:

   --END OF TAPE: CHANGE TAPE AND PRESS <CR>

RESTORE FUNCTION



where:

tape-id is the identity of the tape required by the user (maximum of 6 characters), checked by the system to ensure that the correct tape is mounted.

vol-name is the name of the logical tape volume to be restored to disc (must not be a remote volume, nor "/IPL"); multiple path names (maximum of 10) must be separated by blanks and the list enclosed in quotes.

pathname is the destination path name of the disc volume: it must be the full path name by which the logical disc volume will be known; multiple path names (maximum of 10) must be separated by blanks and the list enclosed in quotes.

occ-number is the number given to the volume when it was dumped to tape, to distinguish it from other possible volumes of the same name.

"R is an option allowing overwriting of a volume already present on disc.

"W is an option requesting the use of WRTALL to send a broadcast message to all connected users, asking them to end their operations, as VOLSR needs a dedicated system. When VOLSR finishes, another WRTALL broadcast informs users that they can resume work.

"RW requests both options.

Characteristics

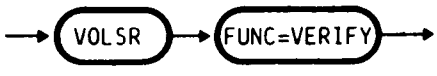
1. Whilst a volume is being restored, the name of the source and destination volumes and their size in bytes is displayed.

2. If restoring a volume requires more than one tape, at the end of each one the following message is displayed, requiring the operator to mount the tape with the appropriate sequence number:

   -- END OF TAPE : INSERT THE TAPE NUMBER <n> AND PRESS <CR>

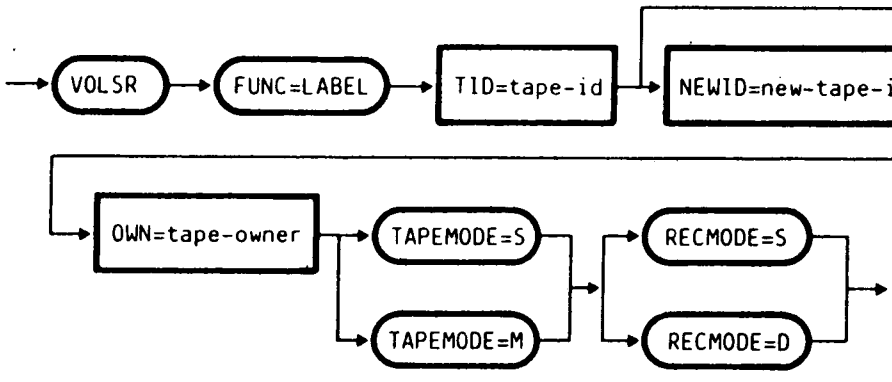
   If the tape inserted has an incorrect sequence number, an error message will be displayed to that effect.

VERIFY FUNCTION



This function does not require parameters.

LABEL FUNCTION



where:

tape-id is the identity of the tape required by the user (contained in the first section of the tape header label, as a maximum of 6 characters), checked by the system to ensure that the correct tape is mounted.

new-tape-id is the new tape identifier, that is the first section of the tape header label to be written (either to a previously unlabelled tape, or overwriting an existing tape header label, in which case the entire tape will be logically deleted).

tape-owner is the name of the tape owner (maximum of 14 characters), which is part of the tape header label.

TAPEMODE=S indicates that there may be several volumes on the tape (multi-volume tape), but none of these can continue on another one.

TAPEMODE=M indicates that the tape is to be dedicated to a single volume, (that is, the tape is single-volume), and that this can continue on other tapes.

RECMODE=S indicates that single-recording will be used for all the volumes written to tape.

RECMODE=D indicates that double-recording will be used for all volumes written to tape, if not SCT5 (SCT5 tapes do not permit double-recording).

Characteristics

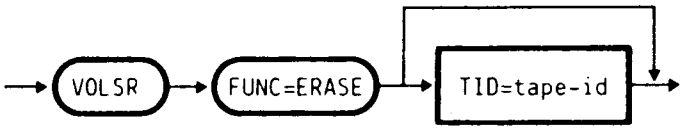
1. If the tape has previously been erased using the ERASE function, the following message appears:

   VIRGIN TAPE

2. If the tape contains data not recorded with VOLSR, the following message appears:

   HEADER TAPE LABEL IS NOT STANDARD

ERASE FUNCTION



where:

tape-id is the identity of the tape required by the user (maximum of 6 characters), checked by the system to ensure that the correct tape is mounted.

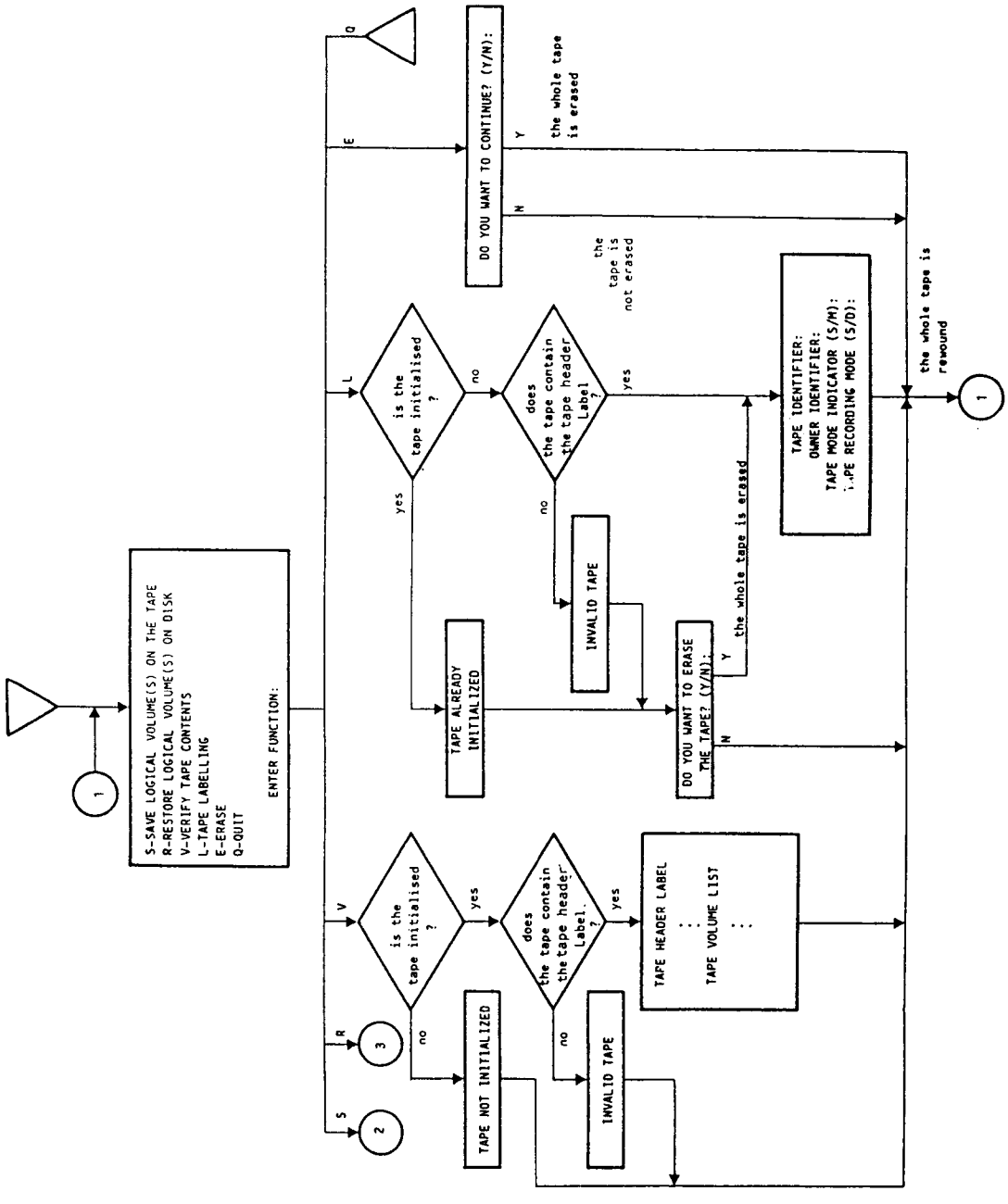


Fig. 3.VOLSR-1 VOLSR Command - Interactive Operator Interface (cont.)

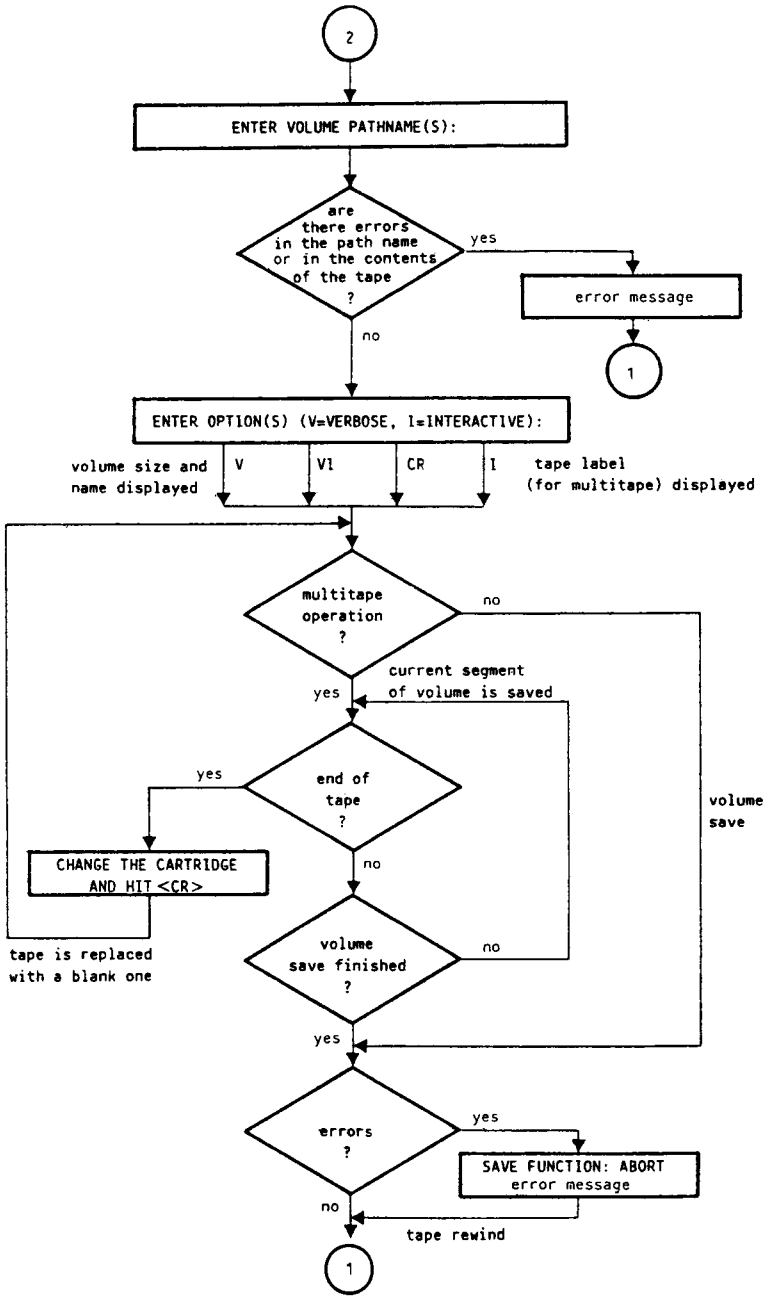


Fig. 3.VOLSR-2 VOLSR Command - Interactive Operator Interface (cont.)

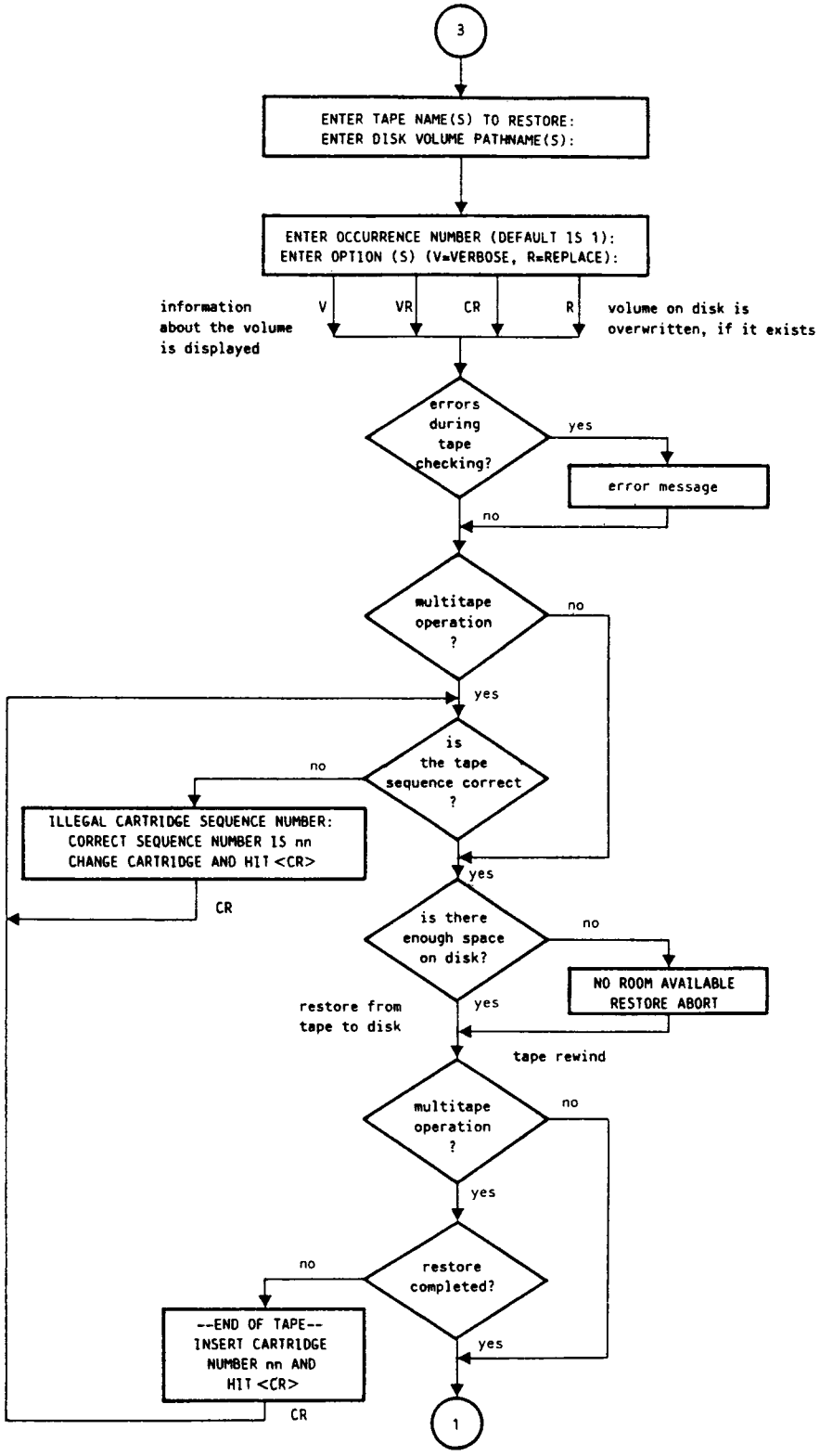


Fig. 3.VOLSR-3 VOLSR Command - Interactive Operator Interface

Initial Display

ENTER FUNCTION:

- S - SAVE LOGICAL VOLUME(S) ON TAPE
- R - RESTORE LOGICAL VOLUME(S) ON DISK
- V - VERIFY TAPE CONTENTS
- L - TAPE LABELLING
- E - ERASE
- Q - QUIT

ENTER FUNCTION :

The user selects the function required, completing the entry with **CR**.

SAVE FUNCTION

ENTER VOLUME PATHNAME(S):

Enter the path name(s) of the volume(s) to save on tape (maximum of 10, separated by blanks). The string length must not exceed 256 characters.

Only logical volumes can be saved, not physical or remote.

xx...x Represents the full volume disc path name. Only the last part of it is used to identify the volume on the tape.

ENTER OPTION(S) (V=VERBOSE, I=INTERACTIVE):

V	Displays name and size of the volume saved (in bytes and sectors). If double-recording has been requested, the numbers of the tracks on which the volume has been saved are also displayed.
I	Displays the tape header label, which is useful for checking multi-tape saves.
VI	Selects both the above options.
CR	Selects neither of the above options.

CHANGE THE CARTRIDGE AND HIT <CR>

The tape being written to is at its end. Mount another SCT, not necessarily initialised (case of a single-volume tape).

CR	Continues the save of the volume on the next SCT.

SAVE FUNCTION: ABORT
error message

A serious error occurred during data transfer from disc to tape: "error message" indicates the nature of the error.

ENTER TAPE NAME(S) TO RESTORE:

Enter the path name(s) of the volume(s) to be restored from tape (maximum of 10, separated by blanks). The path names must be given in exactly the same way as for the SAVE operation, but not necessarily in the same order. The string length must not exceed 256 characters.

Only logical volumes can be restored.

xx...x Represents the tape volume name (maximum of 14 characters).

ENTER DISK VOLUME PATHNAME(S):

Enter the full logical volume path name(s) as they are to appear on disc. The name(s) must not already exist on disc, unless it is intended to overwrite an existing volume. The last segment of each path name must exactly match the corresponding response to the "ENTER TAPE NAME" prompt.

xx...x Represents the full disc path name of the volume.

ENTER OCCURRENCE NUMBER (DEFAULT IS 1):

If only the volume path name to restore is given, this prompt appears to ask for the volume occurrence number: this determines which of the tape volumes of the same name is to be restored.

xx...x Represents the occurrence number of the volume on tape (1 to 99).

ENTER OPTION(S) (V=VERBOSE, R=REPLACE):

V	Displays the names of source and destination volumes, and their size.
R	Allows overwriting of an existing volume on disc.
VR	Selects both options.
CR	Selects neither option.

END OF TAPE: INSERT CARTRIDGE NUMBER nn AND HIT <CR>

The next SCT in the set dedicated to the volume being restored must be mounted: "nn" is the sequence number of the SCT within the set.

CR	Continues restoring from the next SCT.

ILLEGAL CARTRIDGE SEQUENCE NUMBER
CORRECT SEQUENCE NUMBER IS nn
CHANGE CARTRIDGE AND HIT <CR>

The wrong SCT (of a multi-tape volume) has been loaded during a restore operation. Load the correct tape in the device.

CR	Continues the restore function from the correct SCT: the disc volume stays intact.

VERIFY FUNCTION

For the purpose of a user check, this function displays the following:

```
                TAPE  HEADER  LABEL

TAPE IDENTIFIER: xx...x
OWNER IDENTIFIER: xx...x
TAPE MODE: S/M
RECORDING MODE: S/D
CREATION DATE: day month time year

                TAPE  VOLUME(S)  LIST

NAME: volume name
SIZE: nn..n SECTORS, nn...n BYTES
SECTION: sequence number
SAVED ON: day month time year
        .               .               .
NAME: volume name
SIZE: nn...n SECTORS, nn...n BYTES
SECTION: sequence number
SAVED ON: day month time year
        . .             .               .
        .               .               .
        .               .               .

                   END OF VERIFY
```

Note

SECTION will show one number only if verifying a multi-tape volume.

TAPE HEADER LABEL

```
TAPE IDENTIFIER   : . . . . .
OWNER IDENTIFIER  : . . . . .
RECORDING MODE    : . . . . .
CREATION DATE     : . . . . .
```

DO YOU WANT TO ERASE THE TAPE? (Y/N):

This indicates that the SCT mounted has been initialised already (the VOL1 section of the tape header label is present), as shown.

The user is given the chance to confirm that this is the SCT to re-label, as this will erase any data stored on the tape.

If the wrong tape is mounted, reply **N** , dismount the SCT, and the tape will remain intact. A positive reply leads to the prompts given below, with the creation of a new tape header label and effective deletion of the old contents of the SCT.

Y The whole tape is deleted.

N The LABEL function is aborted.

TAPE IDENTIFIER:

Prompts for the tape identifier (maximum of 6 characters) to write at the beginning of the tape header label.

xx...x Represents the tape identifier.

OWNER IDENTIFIER:

Prompts for the name or identifier of the tape owner (maximum of 14 characters), to be added to the tape header label.

xx...x Represents the name or identifier of the tape owner.

TAPE MODE INDICATOR (S/M):

Prompts for the tape mode indicating how the tape is to be used (from single or multi-tape) to be added to the tape header label.

S The tape is to contain several volumes, none of which will be continued on another tape.

M The tape will contain data from a single volume that may be continued on other tapes.

TAPE RECORDING MODE (S/D):

Prompts for the recording mode (from single or double-recording) to be added to the tape header label. (SCT5 does not permit double-recording.)

S Volumes copied from disc to tape will not be duplicated.

D Volumes copied from disc to tape will be duplicated using the double-recording mode facility.

ERASE FUNCTION

DO YOU WANT TO CONTINUE? (Y/N):

The user is asked to confirm that he wants to erase the tape.

Y The whole tape is erased.

N The operation is abandoned.

CANNOT EXECUTE VOLUME DUMP : NO MORE SPACE ON TAPE

The tape specified for the save operation is already full.

CANNOT SAVE VOLUME : NO MORE SPACE ON TAPE

The save operation on tape aborted because the tape is labelled with "TAPEMODE=S" (none of the volumes saved may continue to another tape), and the end of the tape has been reached before the file was completely saved.

ERROR : xx...x IS NOT A VOLUME

The specified object "xx..." is not a volume.

ERROR IN DISK READ : SYSTEM HARDWARE ERROR

Fatal hardware I/O error during a disc read operation.

ERROR IN FILE SEARCH

The tape drive cannot position the tape at the start of the label. This may be because the save operation which wrote the file did not finish successfully, or because of a hardware error.

ERROR IN SEARCH OPERATION

Hardware error during the search for the first free space on tape.

ERROR :INVALID TAPE

The last save operation on tape did not finish successfully.

ERROR IN WRITE OF VOLUME LABEL

Hardware error when writing the volume label on tape.

ILLEGAL PARAMETER NUMBER

The operation requested is not allowed because the number of parameters given does not correspond to the type of the operation.

INCOHERENT PARAMETER NUMBER

The save operation requests that more than one volume must be saved, but the tape is labelled with "TAPEMODE=M" (and must contain a single volume).

INVALID TAPE

The tape does not have a label and must be initialised.

NAME(S) TOO LONG

One of the path names entered has a last section longer than allowed.

PATHNAME IS NOT A VOLUME

The path name specified does not correspond to a volume.

PATHNAME(S) NOT ALLOWED

A save has been requested on a device (like FLn or MFn) not supported by VOLSR.

SAVE FUNCTION ABORT

An error blocking the operation of VOLSR occurred during data transfer between disc and tape.

SAVE PERMISSION DENIED ON OLIVETTI TAPE

This occurs only when using a "slim" drive (of type SCT5) for a save on a tape previously labelled using an SCT4.

TAPE CONTAINS ALREADY A VOLUME

A save has been requested on a tape previously labelled with "TAPEMODE=M" which already contains a volume.

TAPE NOT INITIALIZED

The tape is new or has been erased. The user must initialise the tape before using it.

VOLUME DOES NOT EXIST

The path name of a non-existent volume has been specified for a save operation.

VOLUME NAME TOO LONG

The volume name given in the path name is invalid because longer than 14 characters.

VOLUME PATH NAME TOO LONG

The path name of the volume specified is invalid because longer than 60 characters.

WRONG TAPE IDENTIFIER

The identifier on the tape label does not correspond to that specified by the user.

Restore Error Messages

CANNOT RESTORE : volume-name NO ROOM AVAILABLE

There is not enough disc space for the volume to restore.

CANNOT RESTORE : volume-name VOLUME NOT COMPLETE

The volume has not been restored completely, because the original SAVE did not finish successfully.

CANNOT RESTORE :volume-name NAME ALREADY EXIST

The restore operation of the volume cannot be done because there is already a volume of the same name on the hard disc and the replace option has not been specified.

CONFLICT IN PARAMETER NUMBER

The number of objects to search on tape is different from that to restore to disc.

DO NOT RESTORE THIS VOLUME: IT IS INCOMPLETE

The volume cannot be restored because the previous SAVE did not finish successfully.

ERROR IN CREATION OF: volume-name

A system error occurred during the volume restore.

ERROR : INVALID TAPE

The restore operation cannot be done because the tape was not labelled properly or because the previous save operation did not finish successfully.

ERROR IN VOLUME REMOVE

An error occurred during the deletion from hard disc of a volume to be overwritten by a restore operation.

INCOHERENT VOLUME OCCURRENCE NUMBER

The restore of a single volume on a multi-tape shows an occurence number greater than 1.

INVALID SECTION NUMBER

The tape mounted for the restore operation is not the right one in the set. The tapes must be mounted in the same order as they were used for the save operation.

INVALID TAPE

The tape used does not have a label and must be initialised.

I/O ERROR IN DISK WRITE

The verify function has detected suspect data.

NAME(S) TOO LONG

One of the path names has a last section longer than allowed.

NO ROOM AVAILABLE RESTORE ABORT

There is not enougth space on the hard disc to hold the volume to restore.

STRING CONTENTS "/"

The name entered to be searched on the tape contains a "/" character.

TAPE NOT INITIALIZED

The tape is new or has been erased. The user must initialise the tape before using it.

VOLUME NAME NOT FOUND

The tape does not contain any volume with the specified name.

VOLUME NAME TOO LONG

The volume name given in the path name is invalid because longer than 14 characters.

VOLUME PATH NAME TOO LONG

The path name of the volume specified is invalid because longer than 60 characters.

WRONG TAPE IDENTIFIER

The identifier on the tape label does not correspond to that specified by the user.

Verify Error Messages

ERROR IN FILE SEARCH

The tape drive cannot position the tape at the start of the label. This may be because the save operation which wrote the file did not finish successfully, or because of a hardware error.

INVALID TAPE

The tape does not have a label and must be initialised.

TAPE NOT INITIALIZED

The tape is new or has been erased. The user must initialise the tape before using it.

WARNING : DO NOT RESTORE THIS VOLUME: IT IS INCOMPLETE

The tape contains a volume not saved in its entirety.

WRONG TAPE IDENTIFIER

The identifier on the tape label does not correspond to that specified by the user.

Label Error Messages

ERROR : INVALID TAPE

The tape labelling operation did not finish successfully.

ERROR IN WRITE OF VOLUME LABEL

A hardware error occurred when writing the tape label.

HEADER TAPE LABEL IS NOT STANDARD

The data contained on the tape is not meaningful, as it does not correspond to the standard header structure supported by VOLSR.

TAPE ALREADY INITIALIZED

The tape has already been initialised: it has a label and may contain other data.

TAPE NOT STANDARD

The tape used is not standard.

VIRGIN TAPE

The tape is new or has been erased.

ERASE Error Messages

ERASE UNSUCCESSFUL

The erase operation did not finish successfully.

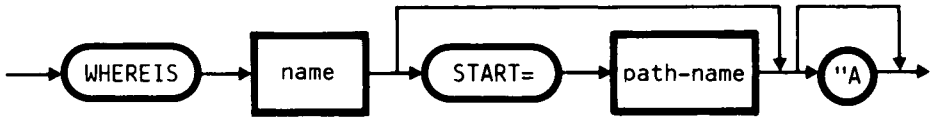
ERROR IN TAPE ERASE

A hardware error occurred during the erase operation on the tape.

Searches for a given file, directory, or volume. If found, its whole path name is displayed.

The command can be directed to begin the search in a particular path by the use of a second parameter.

It is also possible to display the path name of all files, directories, or volumes having the same name.



where:

name is the name of the file, directory or volume whose path name is to be displayed.

path name indicates where to start searching for the file, directory, or volume name. If this is not specified, searching begins from the root ("/").

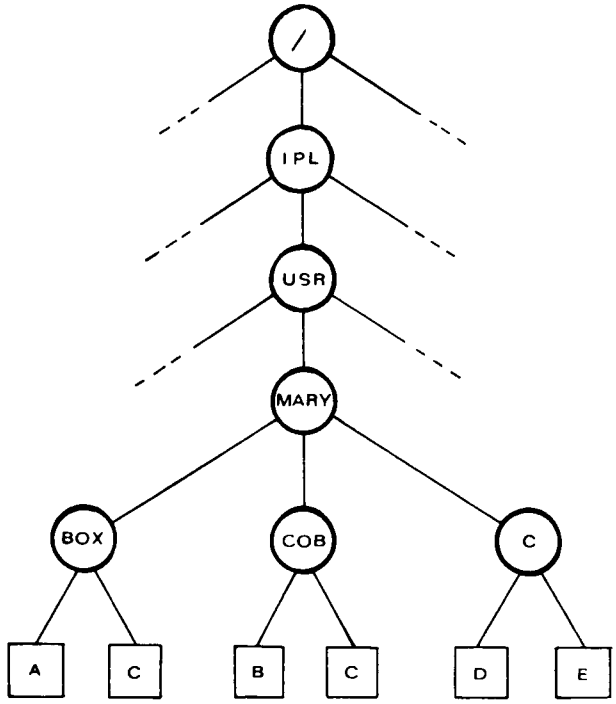
"A is the option requesting display of all the path names of files, directories, or volumes having the name specified. If the option is not used, only the path name of the first occurence found is given.

Characteristics

For executing the command, access rights are required to execute and read the directory from which the search starts, as well as all the included sub-directories.

Examples

With the following memory structure:



---

1. The command

WHERE IS C "A

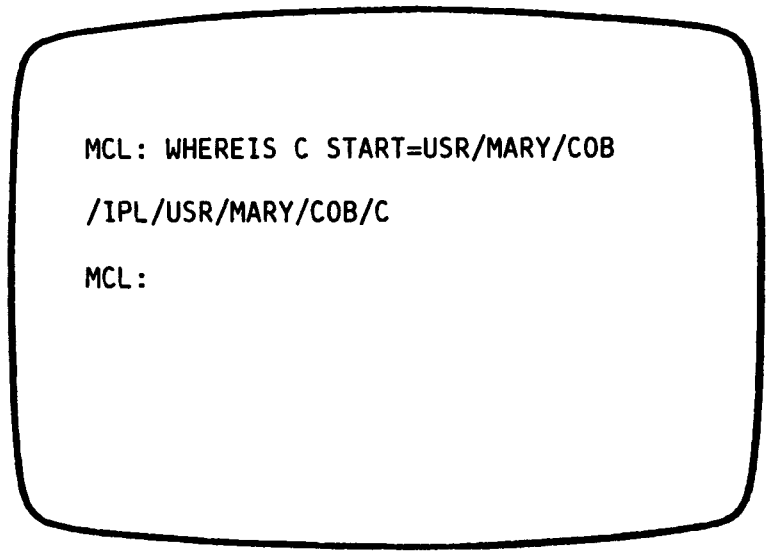
produces the display:

```
MCL: WHEREIS C "A

/IPL/USR/MARY/C
/IPL/USR/MARY/BOX/C
/IPL/USR/MARY/COB/C

MCL:
```

2. The command:

```
WHEREIS C START=USR/MARY/COB
```

produces the display:



For each user currently connected to the system, displays the user name, the name of the terminal to which the user is connected and the date and time of login.

This command has no parameters.

Example

```
MCL: WHO

USER      TTY            LOGIN

ROOT      TTYA     MON OCT 26  07:59:31  1987
GEOFF     TTYB     MON OCT 26  08:34:22  1987
LYNNE     TTYD     MON OCT 26  10:12:11  1987
JOHN      TVC2     MON OCT 26  11:11:48  1987
```

Used to broadcast a message to all users currently connected to the system.

If the command is specified without parameters, it operates in interactive mode. Once the command is entered the Shell prompt is substituted by the '>' character. The user may now type a message of any length. Each line sent (at most 78 characters including carriage return) is displayed as soon as it is typed on the system line of all the users currently connected to the system. The '>' prompt reappears on the sender's screen after a line is sent and the user may type in the next line of the message.

Before transmission of the broadcast message actually starts, the following message is displayed on the screens of all users connected to the system:

BROADCAST MESSAGE FROM user-name

where "user-name" is the name of the user sending the message.

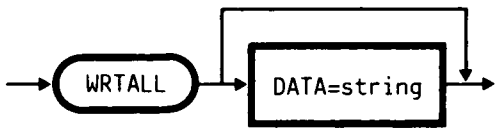
This message as well as each line of the broadcast message is announced by a bell sound.

By pressing **CONTROL D** the user terminates his message and returns to the Shell environment.

The following message is then displayed on the system line of all users connected to the system:

END

Any alphanumeric or special character may be used in the message.



---

where:

string represents one line of the message to send (<= 78 characters, including carriage return).

Characteristics

1. If the message is specified as a parameter in the command line, it is displayed on the system line of the users without being followed by the "END" string.

2. If the message defined in "string" contains blanks, the message must be enclosed in quotes, for instance:

```
MCL: WRTALL DATA='SYSTEM DUMP WILL START IN 5 MINUTES'
```

Used to send a message to a specified user.

If the command is specified without the "DATA" parameter, it operates in interactive mode. Once the command is entered the Shell prompt is substituted by the '>' character. The user may now type a message of any length.

If the receiver specified is actually logged onto the system, each line sent (at most 78 characters including carriage return) is displayed on the system line of his terminal as soon as it is typed. The '>' prompt reappears on the sender's screen after each line is sent and the user may type in the next line of the message.

Before transmission of the message actually starts, the following message is displayed on the screen of the receiver:

MESSAGE FROM user-name

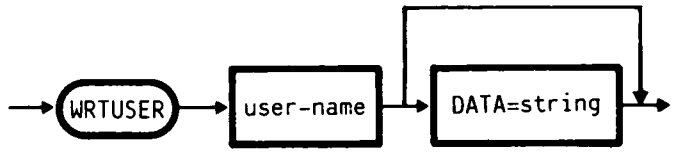
where "user-name" is the name of the sender.

This message as well as each line of the message being received is announced by a bell prompt.

By pressing **CONTROL D** the user terminates the transmission of his message and returns to the Shell environment. The following message is then displayed on the system line of the receiving user:

END

Any alphanumeric or special character may be used in the message.



where:

user-name is the user name of the receiver.

string represents one line of the message to be sent (<=78 characters including carriage return).

Characteristics

1. If the receiving user is not logged onto the system, the following message is displayed on the screen of the sender:

   WARNING: user-name NOT LOGGED

   where "user-name" is the name of the receiver. In this case, each line of the message being sent is stored in the .MESG file of the receiver.

2. If the message is specified in the parameter of the command, it is displayed on the system line of the user without being followed by the "END" string.

3. If the message defined in "string" contains blanks, the message must be in quotes, for instance:

   MCL: WRTUSER JOHN DATA='RESERVED FILES'

Gives a physical description of any file system object on disc or tape, in a recursive manner if required.

where:

path-name is the path name of the file to investigate. This may refer to any file system object, but only a single path name is permitted.

"R displays the contents of the path name "recursively", so that any "nested" directories within the path name are displayed also. The occurence of a nested directory at one level is followed by the display of its contents.

If **"R** is not specified, nested directories are not detailed.

"W displays the number of free extents on the volume identified by the path name, according to the volume's PDD TABLE information.

DISPLAY

The information displayed is as follows:

NAMEFILE is the file name of every file on the volume.

PDD is the PDD (Permanent Dataset Descriptor) number of the file, as recorded in the PDD TABLE.

LENGTH is the amount of data in the file (in bytes).

ALLOC-UNIT is the allocation unit (in bytes): amount of contiguous space allocated to the file whenever expansion is necessary.

LINK is the number of linked files.

EXT is the number of extents belonging to the file.

EXTENSION is the amount of disc space allocated to the file (in bytes). This is always 0 for tape files.

TYPE shows simple file types, as stored in the PDD field, from: DEV (device), DIR (directory), VOL (volume), PRGDIR (program directory), BST (byte-stream), ALIAS (alias), but not complex types such as keyed or positional.

If "W has been specified, the amount of free extents in the volume containing the specified path name (as indicated in the PDD table of the volume) is displayed in the following message:

NUMBER OF FREE EXTENTS := xxxxx

Characteristics

1. WSTAT only functions in non-interactive mode.

2. Nested directories as well as the directory referred to directly can be displayed.

3. Whilst references made by a disc PDD to other PDDs on the disc may be followed, only one tape PDD may be read, therefore the tape is considered as a single object regardless of its real contents.

4. The access rights required to execute this command are read and execute on the volume, the PDD table, and all the intermediate file system objects comprised between the volume and the object specified in the path name.

PART 3 - RESERVED COMMANDS

INTRODUCTION TO PART 3

This part (chapter 4) describes the commands that are only available to the system administrator (user name ROOT), in alphabetical order.

The method used to describe the command syntax is specified in chapter 3 of the DOCUMENTATION Guide.

4. RESERVED COMMANDS DESCRIPTION

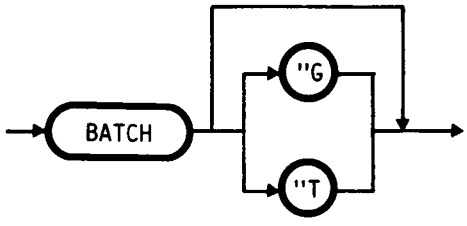
INTRODUCTION

This chapter contains details of commands which are reserved for the use of system administrators.

These commands allow the system administrator to control the allocation of physical resources within the system and to organise the operations of the users.

The system administrator may control the batch environment using the following reserved functions:

- start the batch class (jobs in the class will be executed)
- stop the batch class (jobs in the class will not be executed)
- change the priority or status of, or kill a job submitted by a normal user.



where:

"G is the option which allows the management of the jobs belonging to the global batch class in a distributed system.

"T is the option which allows the management of the jobs to be performed by a specified time.

The function types and the parameters are entered interactively.

The system administrator is presented with the following menu:

```
                    BATCH SYSTEM
                   --------------

     DO YOU WANT TO:

             L = LIST THE BATCH CLASS
             U = STOP THE BATCH CLASS
             M = START THE BATCH CLASS
             J = SHOW JOB STATUS
             H = SET HOLD A JOB
             R = SET READY A JOB
             C = CHANGE PRIORITY OF A JOB
             K = KILL A JOB
             E = END

                 NEXT CHOICE >_
```

Fig. 4.BATCH-1 BATCH COMMAND - System Administrator Menu

OPERATION

The operation of the menu is described under BATCH in Part 2.

Parameters

The parameters required by the functions available to the system administrator are shown in the following table:

BATCH FUNCTION	COMMAND LETTER	PARAMETERS REQUIRED parameter1	parameter2
list the class	L	-	-
stop the class	U	-	-
start the class	M	-	-
show job status	J	job number	-
set hold a job	H	job number	-
set ready a job	R	job number	-
change priority of a job	C	job number	priority
kill a job	K	job number	-

Tab 1. BATCH Functions Parameters

The parameter ranges acceptable are as follows:

- Job number: an integer in the range 1 to 63. This is allocated when the job is submitted.

- Priority: an integer in the range 1 to 127, where a job with the lowest priority number will be executed first. Normally jobs are given a priority of 32. The system administrator may change the priority of a job at any time. Jobs of equal priority are executed in the order in which they were submitted.

Parameters may be input after the function letter. For example, to change the priority of job 12 to 1, the system administrator may enter:

```
C 12 1
```

It is necessary to insert spaces between the parameters.

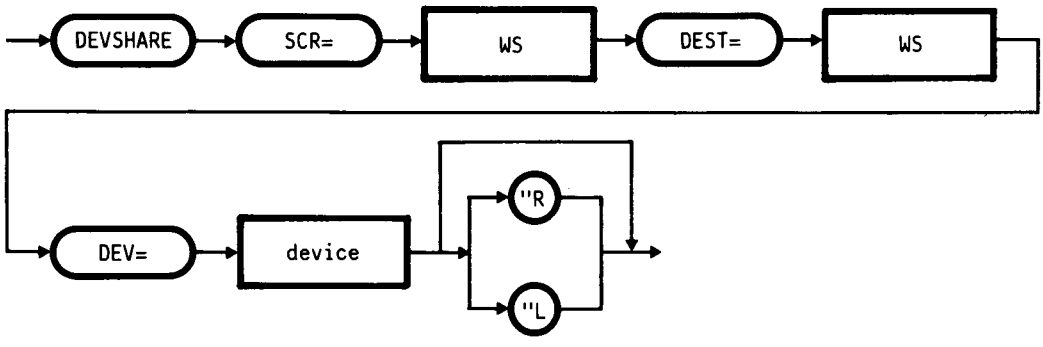
The following message may be presented to the system administrator in addition to those described under BATCH in Part 2:

```
PRIORITY (1-127) =
```

nnn Specifies the new priority of the selected job.

. Returns to the main menu.

Allows a printer or cash adapter, configured for one workstation, to be shared by another workstation.



where:

SRC ws is the name of the workstation for which the printer or cash adapter is configured (**ws** = TTYX or VTX1, VTX2, VTX3, VTX4 with X = A, B,... or 0..5).

DEST ws is the name of the workstation to which the printer or cash adapter must be assigned (**ws** = TTYX or VTX1, VTX2, VTX3,VTX4 with X = A, B,... or 0..5)

device is the peripheral to assign to the workstation. This parameter can assume one of the following values:

PR if the peripheral is a printer

CA if the peripheral is a cash adapter.

"R is an option indicating the right position of the dispenser slot.

"L is an option indicating the left position of the dispenser slot.

Characteristics

1. Two cash adapters cannot be assigned to the same workstation.

2. If the peripheral is a cash adapter, and the same workstation is specified in the parameters SRC and DEST, the position option must be specified.

Error Messages

ERROR IN PARAMETERS

An error has been made in specifying a command parameter.

INVALID OPERATION

Two cash adapters cannot be assigned to the same workstation.

INVALID OPTION

An incorrect command option has been specified.

UNIT NOT AVAILABLE

The peripheral specified is not available.

Installs and uninstalls any MOS package.

The parameters and the type of the operation to perform are defined interactively.

Characteristics

1. INSTALPKG is a general installation procedure, which does not depend on the software package to be installed in any way.

2. The package to be installed may be stored on floppy discs, mini-floppies, SCT, or MTU. For floppy discs and mini-floppies INSTALPKG loads the volume /INSTVOL under the directory "/"; for SCT and MTU INSTALPKG copies the streaming cartridge tape or magnetic tape contents under /IPL/INSTVOL, by means of the utilities VOLSR and FILETAR respectively.

3. INSTALPKG only works in interactive mode.

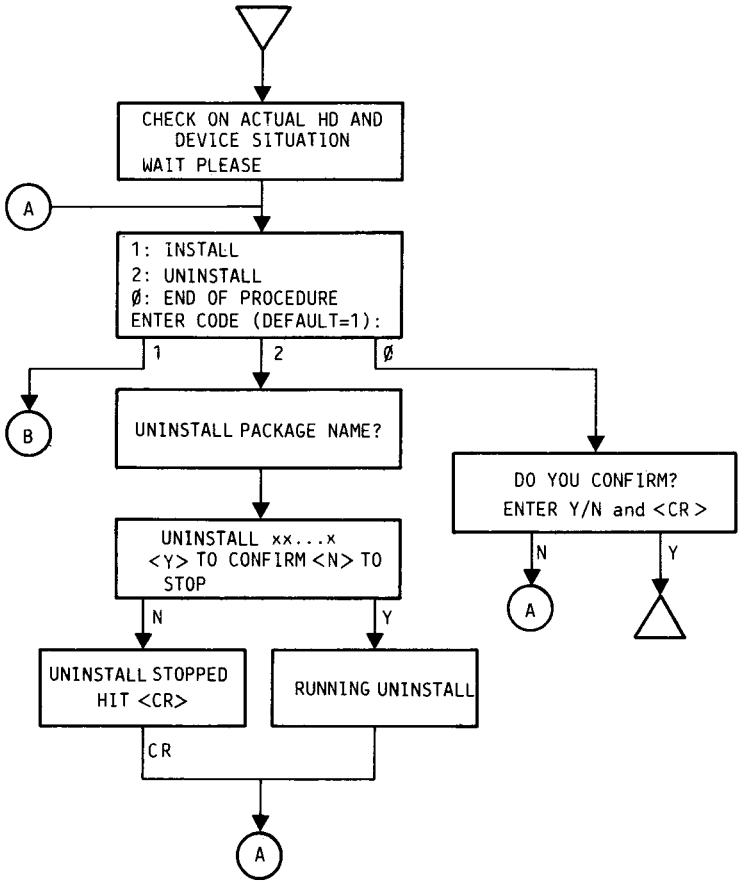


Fig. 4.INSTALPKG-1 INSTALPK Command - Operator Interface (cont.)

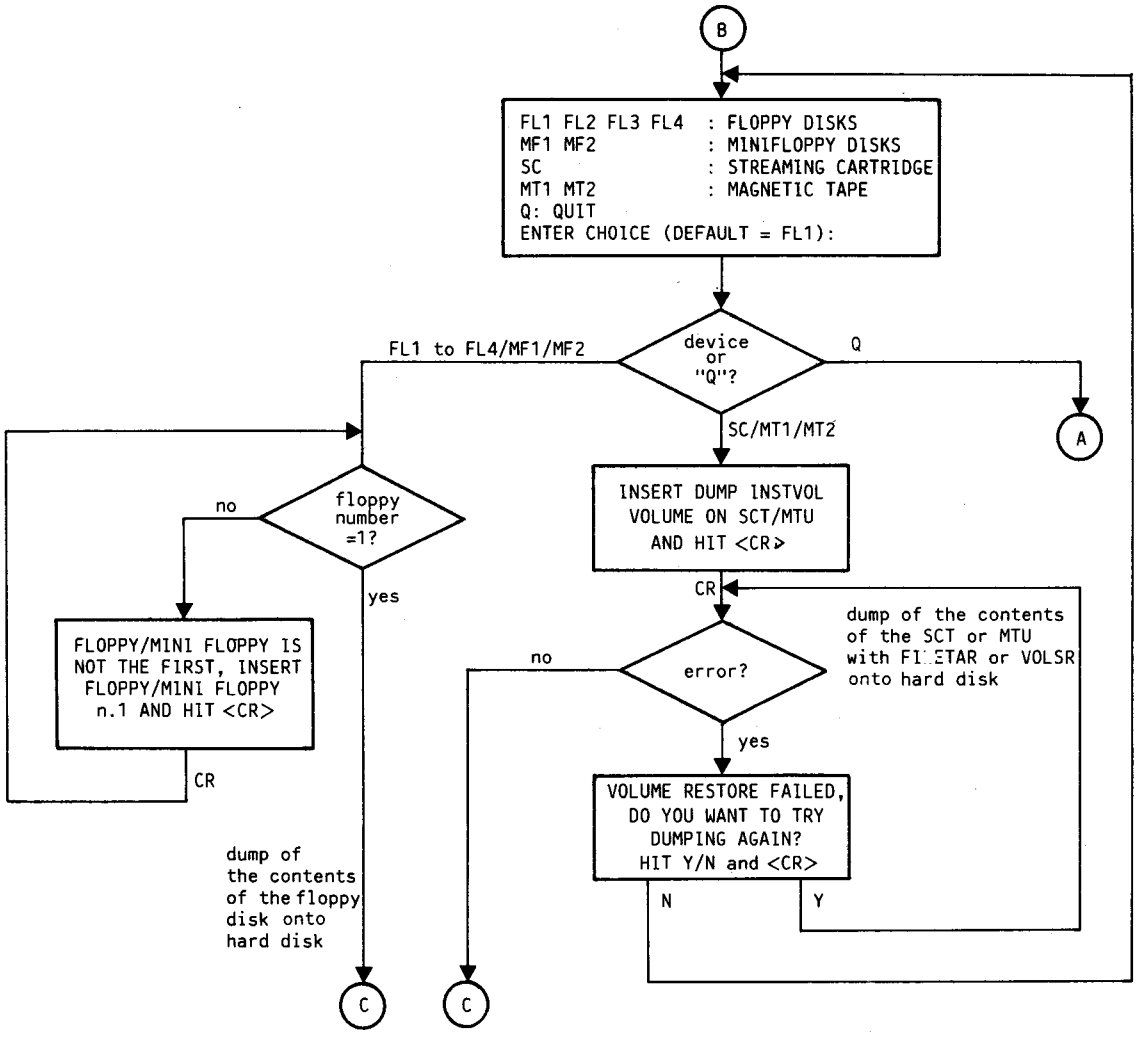


Fig. 4.INSTALPKG-2 INSTALPK Command - Operator Interface (cont.)

CHECK ON ACTUAL HD AND DEVICE SITUATION
WAIT PLEASE

As soon as INSTALPKG is active, it checks the current hard disc state.

The operator can select the type of operation to perform from the main menu:

1: INSTALL
2: UNINSTALL
0: END OF PROCEDURE

ENTER CODE (DEFAULT = 1):

1	Installs a package.
2	Removes an installed package.
0	Exits from the INSTALPKG procedure.
CR	Enters the default install option (1).

DO YOU CONFIRM?
ENTER Y/N AND <CR>

When the user chooses to exit from the procedure, this message appears to prompt for confirmation, as follows:

Y	Leaves the procedure.
N	Returns to the main menu.

The user is asked to specify the magnetic support on which the package is stored, in the following menu:

```
FL1 FL2 FL3 FL4              : FLOPPY DISKS
MF1 MF2                      : MINI FLOPPY DISKS
SC                           : STREAMING CARTRIDGE
MT1 MT2                      : MAGNETIC TAPE
Q                            : QUIT

ENTER CHOICE (DEFAULT=FL1):
```

FLn	The package is on the floppy disc in drive "n".
MFn	The package is on the mini-floppy disc in drive "n".
SC	The package is on streaming cartridge tape.
MTn	The package is on magnetic tape "n".
CR	The default value FL1 is entered.

FLOPPY/MINI FLOPPY IS NOT THE FIRST, INSERT FLOPPY/MINI FLOPPY n.1 AND HIT <CR>

If the package is stored on several floppy discs or mini-floppies, INSTALPKG first checks the order, and then asks the user to insert the first floppy disc or mini-floppy in the drive.

INSERT DUMP INSTVOL VOLUME ON SCT/MTU AND HIT <CR>

If the package is on SCT or MTU, the user is asked to insert the unit in the appropriate drive, so as to copy the volume onto hard disc.

VOLUME RESTORE FAILED, DO YOU WANT TO TRY DUMPING AGAIN?
HIT Y/N AND <CR>

An error has occurred during the dump from SCT/MTU to hard disc. The user is asked whether another attempt should be made.

Y Attempts another copy operation.

N Returns to the menu asking for the support on which the package is stored.

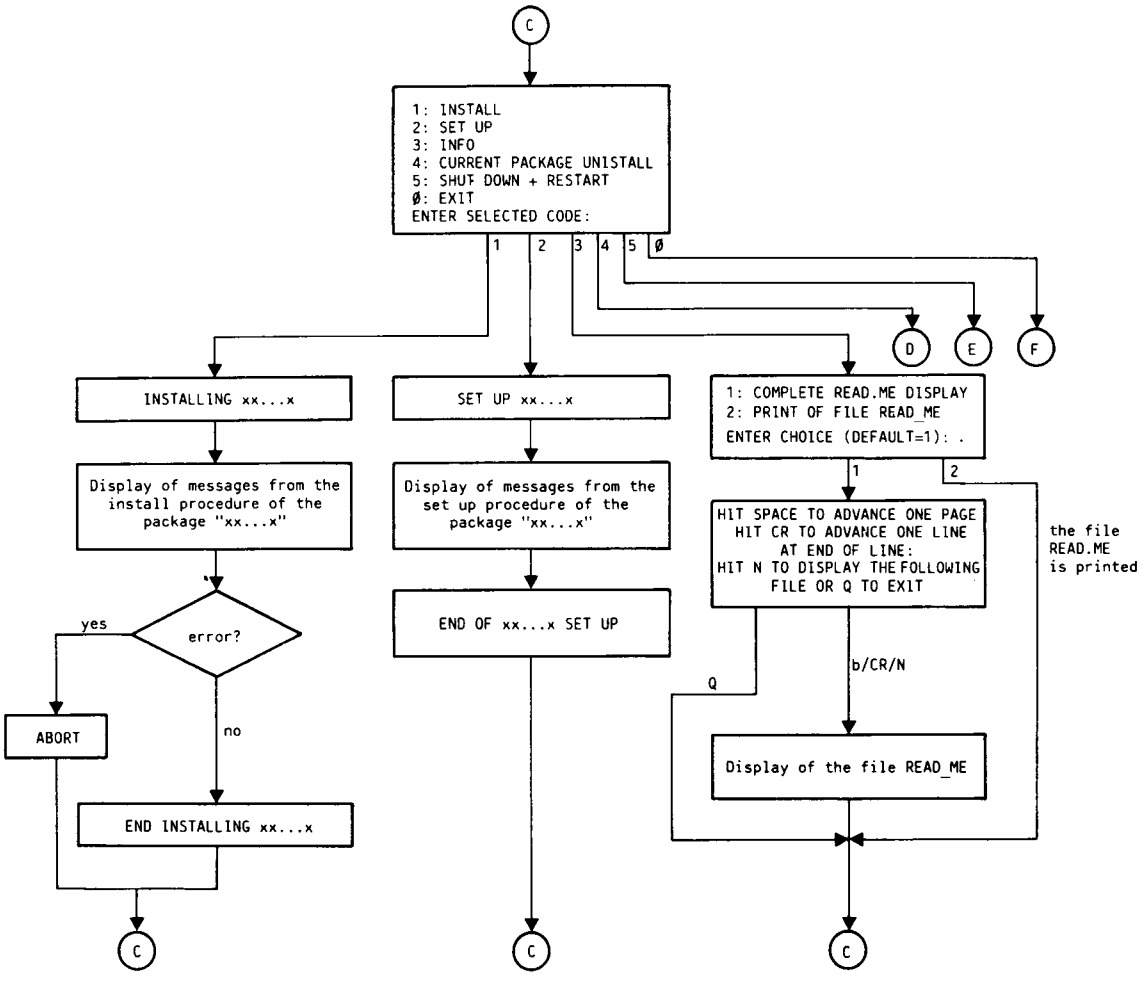


Fig. 4.INSTALPKG-3 INSTALPK Command - Operator Interface (cont.)

In the next menu, the user is asked what kind of operation is to be performed:

```
1: INSTALL
2: SETUP
3: INFO
4: CURRENT PACKAGE UNINSTALL
5: SHUTDOWN + RESTART
0: EXIT

ENTER SELECTED CODE:
```

1	The package installation procedure is started.
2	The package SETUP procedure is started, to make any changes or updates required in the operation of the package itself.
3	Information about the package is displayed.
4	The procedure for uninstalling the package is started, in the case where the installation just attempted did not finished properly.
5	The system is shut down and restarted.
0	The installation procedure is abandoned.
CR	Completes the entry.

```
ATTENTION, IF EXIT CHOICE IS CONFIRMED, TEMPORARY VOLUME WILL BE
UNMOUNTED OR REMOVED.
DO YOU CONFIRM? Y/N AND <CR>
```

The user is asked to confirm that exit from the program is required, as the temporary volume will be deleted if the answer is yes.

Y	Exits from the installation procedure and returns to the main menu.
N	Returns to the installation procedure menu.

The following menu is displayed when the user asks for information on the package installed or to be installed.

```
1: COMPLETE READ_ME DISPLAY
2: PRINT OF FILE READ_ME

ENTER CHOICE (DEFAULT=1):
```

1	Displays the data in the READ_ME file.
2	Prints the data in the READ_ME file on the system printer SYSPRT1.
CR	Selects the default value 1.

```
HIT SPACE TO ADVANCE ONE PAGE
HIT CR TO ADVANCE ONE LINE
AT END OF FILE:
HIT N TO DISPLAY THE FOLLOWING FILE OR Q TO EXIT.
```

When the file containing information on the package is displayed, the user can go on to the next screen page by pressing the space bar, or go on to the next line by pressing **CR**. At the end of the information display, the user can press **N** to display the next file, or **Q** to leave this menu and return to the installation procedure menu.

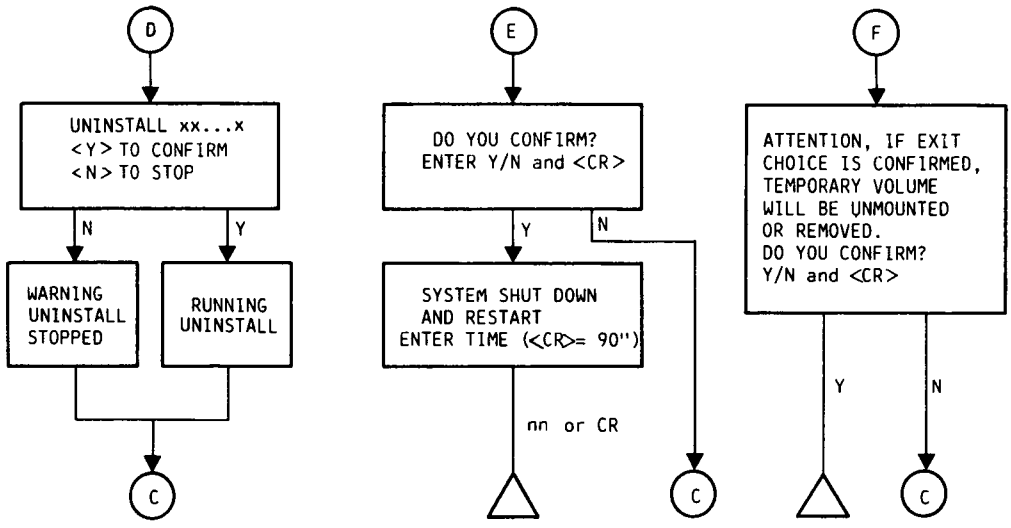


Fig. 4.INSTALPKG-4 INSTALPK Command - Operator Interface

UNINSTALL xx...x . <Y> TO CONFIRM <N> TO STOP.

The user is asked to confirm whether to uninstall package xx...x, after an installation procedure abort.

Y The UNINSTALL package procedure is performed.

N The procedure is not performed.

DO YOU CONFIRM ? ENTER Y/N AND <CR>

The user is asked to confirm whether the system is to be shutdown and restarted.

Y Confirms the system shutdown and restart.

N Returns to the installation procedure.

SYSTEM SHUTDOWN AND RESTART
ENTER TIME (<CR> = 90"):

The user is asked to enter the time in seconds before the system is to be shutdown.

nn Represents the time before the shutdown.

CR Entered on its own, selects the default value of 90 seconds.

Description of the Uninstall Procedure

UNINSTALL PACKAGE NAME ?

Prompts for the name of the package to be uninstalled.

xx...x Represents the name of the package to be uninstalled.

UNINSTALL xx...x <Y> TO CONFIRM <N> TO STOP

Prompts the user to confirm that package "xx...x" is to be uninstalled.

Y Uninstalls the package.

N Cancels the uninstall procedure request.

INSTALL NOT AVAILABLE

The INSTALL procedure contained in the package is not available.

INVALID OPERATION

The floppy or mini-floppy disc containing the package has been inserted in the drive already.

PKGNAME NOT AVAILABLE

The UNINSTALL package procedure is not available, as the last installation did not complete successfully.

PROCEDURE ABORT
INSTALPKG CAN BE ACTIVATED ONLY BY ROOT

INSTALPKG can only be activated by the system administrator.

READ_ME NOT AVAILABLE

The READ_ME file containing information on the package to install is not available.

SETUP NOT AVAILABLE

The SETUP procedure contained in the package is not available.

SHUTDOWN+RESTART NOT AVAILABLE

The procedure RESTART has not been configured.

PKGNAME UNINSTALL PROCEDURE NOT AVAILABLE
CHECK ON THIS LIST THE CORRECT NAME OF PACKAGE UNINSTALL PROCEDURE

The UNINSTALL procedure contained in the package "PKGNAME" is not available. A list containing the names of the uninstall procedures available is displayed.

ENTER UNINSTALLING PACKAGE NAME WITHOUT SUFFIX AND HIT <CR>:

This prompts the user to enter the name of the uninstall procedure, without its suffix. If this procedure is not found, the following message appears:

PKGNAME NAME NOT FOUND

and control returns to the main menu.

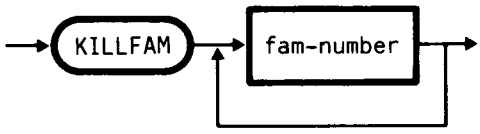
WARNING: GETLINE (MRFL) NOT FOUND UNDER /IPL/DPC/CMD
ENTER NEW PATHNAME:

The support files GETLINE and MRFL needed by the utility are not found in the directory /IPL/DPC/CMD. The user is prompted to enter the path name of the directory containing these files.

WARNING SYSTEM NOT CORRECTLY GENERATED
path name DOES NOT EXIST
HIT <CR> TO CONTINUE

During the hard disc checking phase, the system checks the existence of a few files and directories defined in "path name", which are needed by the procedure contained in the package to install. The absence of these files or directories does not cause the utility to abort, but the user is advised to exit from it from the main menu, and to tidy up the disc situation.

This privileged command is used to deactivate one or more families executed interactively or in background.



where:

fam-number is the number identifying the family.

Characteristics

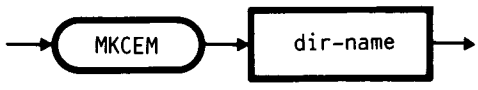
1. The SHFAM command must be used to find out the identifier number of the family to deactivate.

2. After a family has been deactivated, the system displays:

   *** FAMILY n KILLED ***

Enables the management of the data base of system messages. The functions provided by MKCEM operate on the following items:

- CEMDB work data base
- installed data base $CEM
- files of the classes containing the messages.



---

where:

dir-name is the name or path name of the work directory containing the data base CEMDB and the class files.

The operation to perform is selected interactively from the following main menu:

```
        1: WORK-DIR DATA BASE MANAGEMENT

        2: INSTALLED DATA BASE MANAGEMENT

        3: FILE CLASS MANAGEMENT

        4: QUIT

        ENTER YOUR CHOICE >
```

Characteristics

1. After the data base $CEM is updated and installed, the system must be reset and then restarted in order to be able to use it.

2. Each system component is assigned a class containing the error codes and associated messages specific to the component. The class is identified by a decimal value.

3. When a class is created, a byte-stream file is created in the work directory; the name of the file is represented by "CExxxx", where "xxxx" is the name of the class.

4. The system and date-related messages are in class 16.

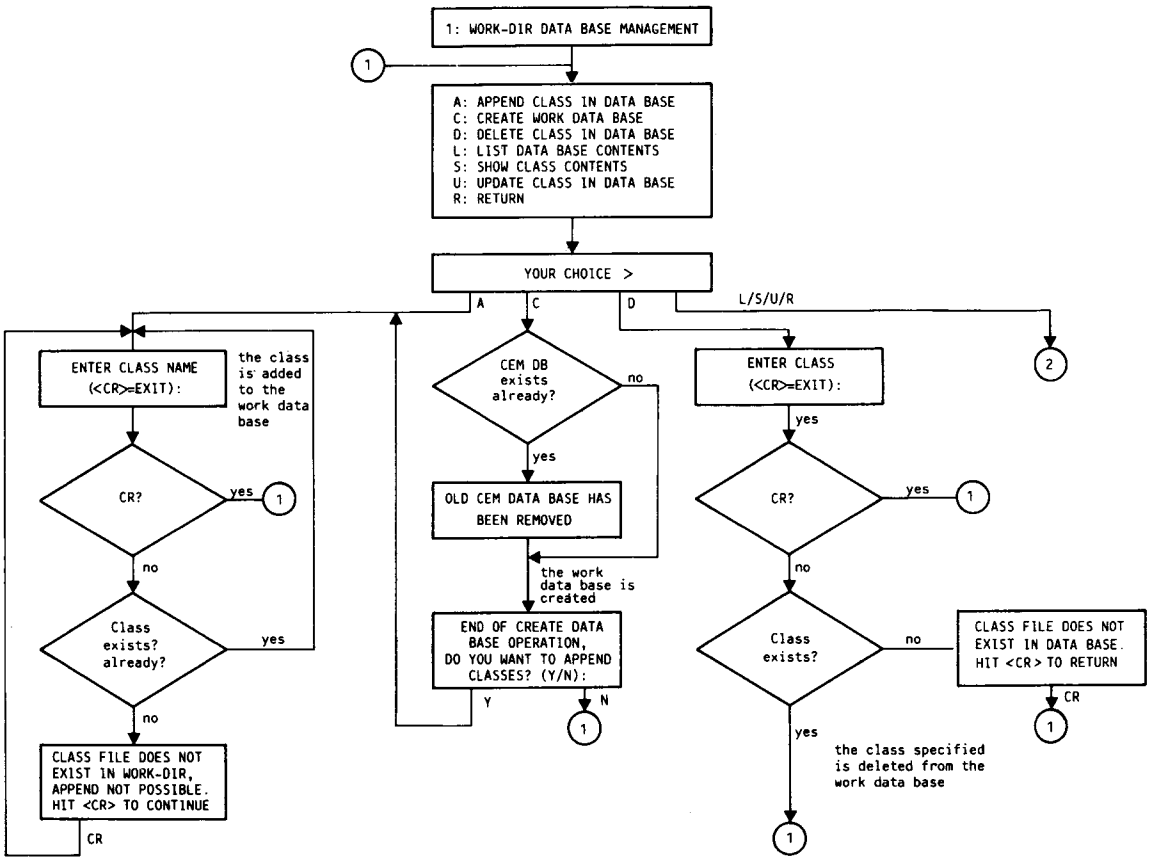


Fig. 4.MKCEM-1 MKCEM Command - Operator Interface (cont.)

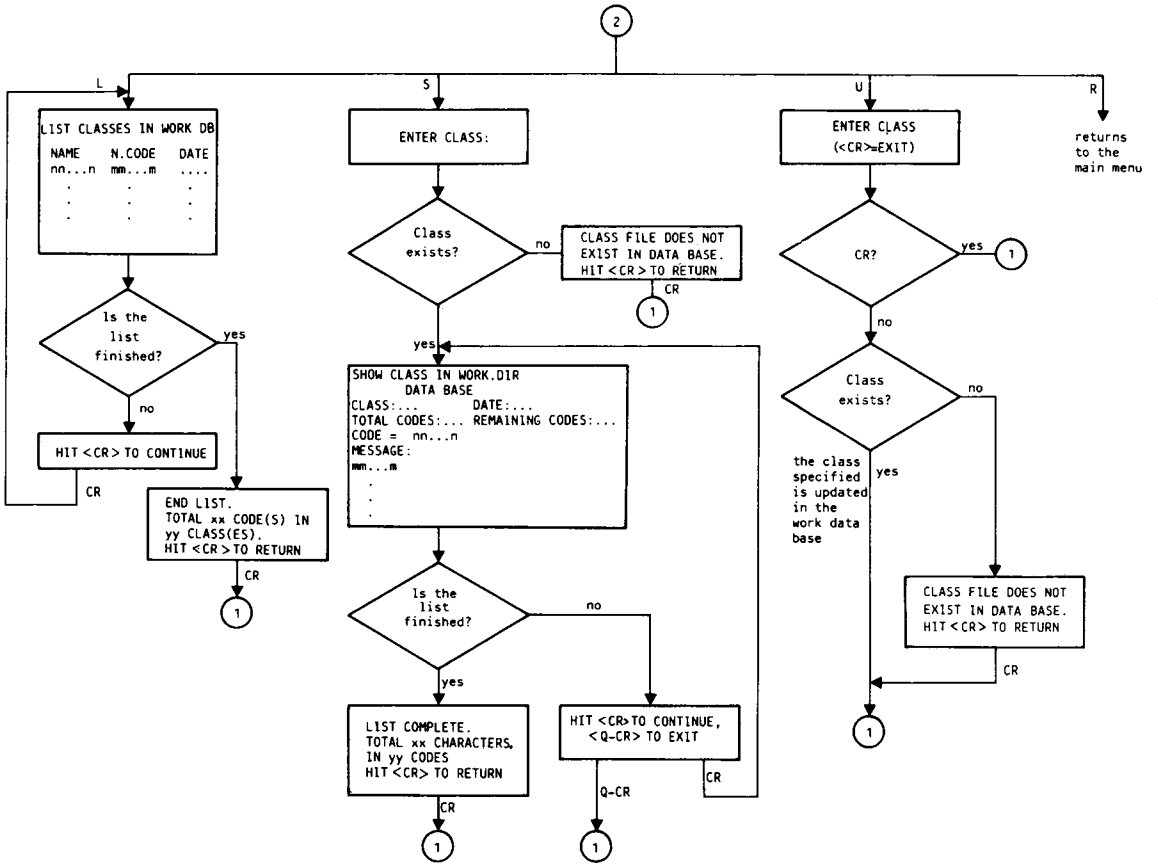


Fig. 4.MKCEM-2 MKCEM Command - Operator Interface (cont.)

The first choice in the main menu allows the management of the work data base CEMDB with the following operations:

```
A: APPEND CLASS IN DATA BASE
C: CREATE WORK DATA BASE
D: DELETE CLASS IN DATA BASE
L: LIST DATA BASE CONTENTS
S: SHOW CLASS CONTENTS
U: UPDATE CLASS IN DATA BASE
R: RETURN

YOUR CHOICE >
```

A	Adds a class to the work data base.
C	Creates a new work data base.
D	Deletes a class from the work data base.
L	Displays the classes contained in the work data base.
S	Displays the messages in the class specified.
U	Updates the specified class in the data base.
R	Returns to the main menu.

```
ENTER CLASS NAME (<CR>=EXIT):
```

Prompts for the class name.

CR	Returns to the menu.

```
CLASS FILE DOES NOT EXIST IN WORK-DIR, APPEND NOT POSSIBLE.
HIT <CR> TO CONTINUE
```

The specified class cannot be added to the work data base, as the associated file is not in the directory.

CR	Returns to the class name prompt.

OLD CEM DATA BASE HAS BEEN REMOVED

The data base CEMDB of the work directory has been removed because creation of a new data base has been requested.

END OF CREATE DATA BASE OPERATION, DO YOU WANT TO APPEND CLASSES? (Y/N)

The creation of the new data base has completed successfully. The user is now asked whether any classes should be added to the new data base created.

Y To add new classes to the new data base.

N If no more classes are to be added to the new data base.

CLASS FILE DOES NOT EXIST IN DATA BASE, HIT <CR> TO RETURN

The class file specified for display, update, or deletion is not in the work data base. Press **CR** to return to the menu.

LIST CLASSES IN WORK DB

```
NAME          N.CODE          DATE
nn...n        mm...m          ......
  .             .               .
  .             .               .
  .             .               .
```

The following information is displayed for each class in the work data base:

- the name (NAME)
- the number of messages contained (N.CODE)
- the date of creation (DATE).

END LIST. TOTAL xx CODES IN yy CLASS(ES). HIT <CR> TO RETURN

At the end of the list the total number of messages ("xx") and classes ("yy") in the work data base is displayed. Press **CR** to return to the menu.

HIT <CR> TO CONTINUE

If the information to be displayed fills more than one screen page, the user may press **CR** to display the next screen page.

SHOW CLASS IN WORK-DIR DATA BASE

```
CLASS : ......                  DATE : ......
TOTAL CODES : ......            REMAINING CODES : ......

CODE = nn...n
MESSAGE :
mm...m
   .
   .
   .
```

When requesting the display of the contents of a class, the information obtained is as follows:

- codes and corresponding messages
- class name (CLASS)
- date of creation (DATE)
- total number of codes (TOTAL CODES)
- the number of codes still to display (REMAINING CODES).

LIST COMPLETE. TOTAL xx CHARACTERS IN yy CODE(S). HIT <CR> TO RETURN

At the end of display, the total number of characters in the messages ("xx") and the total number of codes ("yy") in the class is given. Press **CR** to return to the menu.

HIT <CR> TO CONTINUE, <Q - CR> TO EXIT

If the information to be displayed fills more than one screen page, the user may press **CR** to display the next screen page, or **Q** followed by **CR** to interrupt the display and return to the menu.

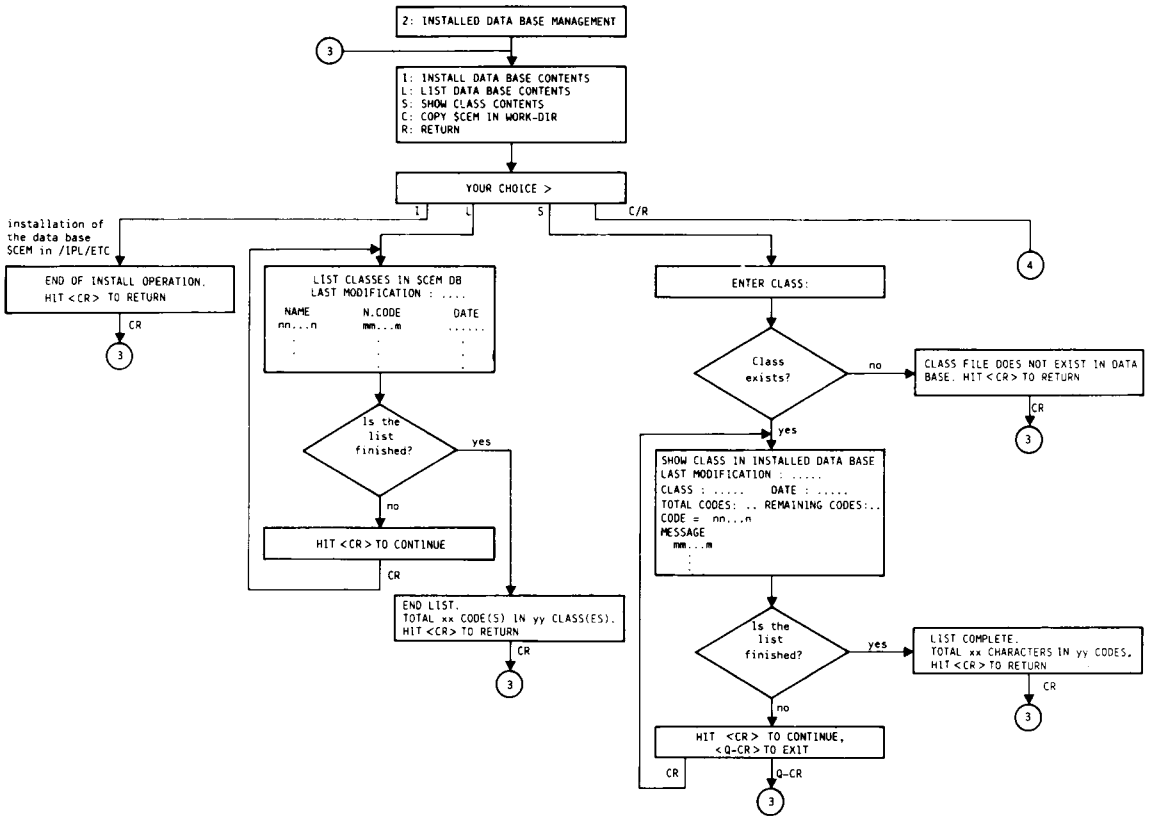


Fig. 4.MKCEM-3 MKCEM Command - Operator Interface (cont.)

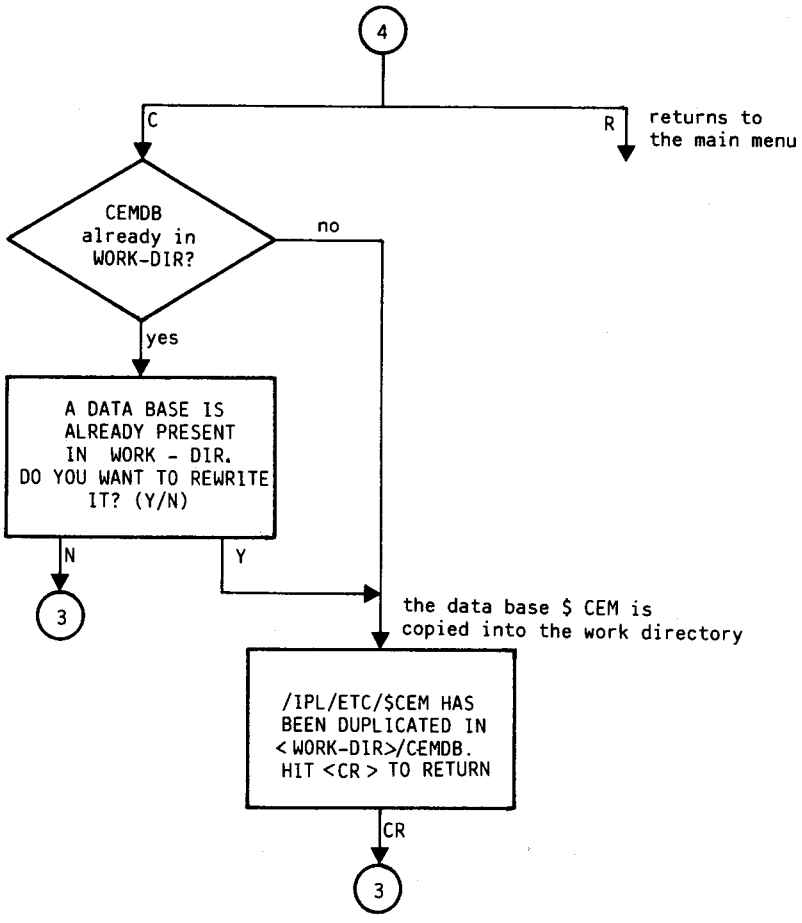


Fig. 4.MKCEM-4 MKCEM Command - Operator Interface (cont.)

The second choice in the main menu allows the management of the installed data base $CEM with the following operations:

```
I: INSTALL DATA BASE CONTENTS
L: LIST DATA BASE CONTENTS
S: SHOW CLASS CONTENTS
C: COPY $CEM IN WORK-DIR
R: RETURN

YOUR CHOICE >
```

I	Installs the work data base CEMDB under /IPL/ETC/$CEM.
L	Displays the classes of the installed data base.
S	Displays the codes and messages in the specified class of the $CEM data base.
C	Copies the installed data base $CEM into the work directory, as CEMDB.
R	Returns to the main menu.

```
END OF INSTALL OPERATION. HIT <CR> TO RETURN
```

Installation of the data base $CEM has ended. Press **CR** to return to the menu.

```
LIST CLASS IN $CEM DB        LAST MODIFICATION : ......

NAME            N.CODE              DATE
nn...n          mm...m              ......
  .               .                   .
  .               .                   .
  .               .                   .
```

The following information is displayed for each of the classes in the installed data base:

- the name (NAME)
- the number of messages contained (N.CODE)
- the date of creation (DATE).

The date of last modification of the data base is also displayed.

HIT <CR> TO CONTINUE

If display of the information covers more than one screen page, the user may press **CR** to display the next screen page.

END LIST. TOTAL xx CODES IN yy CLASS(ES). HIT <CR> TO RETURN

At the end of the list, the total number of messages ("xx") and classes ("yy") in the installed data base $CEM is displayed. Press **CR** to return to the menu.

ENTER CLASS:

Prompts for the class name.

```
SHOW CLASS IN INSTALLED DATA BASE      LAST MODIFICATION : ......

CLASS : ......                         DATE : ......
TOTAL CODES : ......                   REMAINING CODES : ......

CODE = nn...n
MESSAGE :
mm...m
  .
  .
  .
```

When requesting the display of the contents of a class, the information obtained is as follows:

- codes and corresponding messages
- class name (CLASS)
- date created (DATE)
- total number of codes (TOTAL CODES)
- the number of codes still to display (REMAINING CODES).

The date of last modification of the class is also displayed.

HIT <CR> TO CONTINUE, <Q - CR> TO EXIT

If display of the information covers more than one screen page, the user may press **CR** to display the next screen page, and **Q** followed by **CR** to interrupt the display and return to the menu.

CLASS FILE DOES NOT EXIST IN DATA BASE. HIT <CR> TO RETURN

The class whose codes and messages are to be displayed is not in the data base $CEM. Press **CR** to return to the menu.

LIST COMPLETE. TOTAL xx CHARACTERS IN yy CODE(S). HIT <CR> TO RETURN

At the end of display, the total number of characters in the messages ("xx") and the total number of codes ("yy") in the class is given. Press **CR** to return to the menu.

A DATA BASE IS ALREADY PRESENT IN WORK-DIR.
DO YOU WANT TO REWRITE IT? (Y/N)

A data base CEMDB is already present in the working directory; the user is asked whether it should be overwritten.

Y To overwrite the data base in the work directory.

N To return to the menu.

/IPL/ETC/$CEM HAS BEEN DUPLICATED IN <WORK-DIR>/CEMDB. HIT <CR> TO RETURN

The data base $CEM under /IPL/ETC has been copied into the work data base CEMDB in the work directory. Press **CR** to return to the menu.

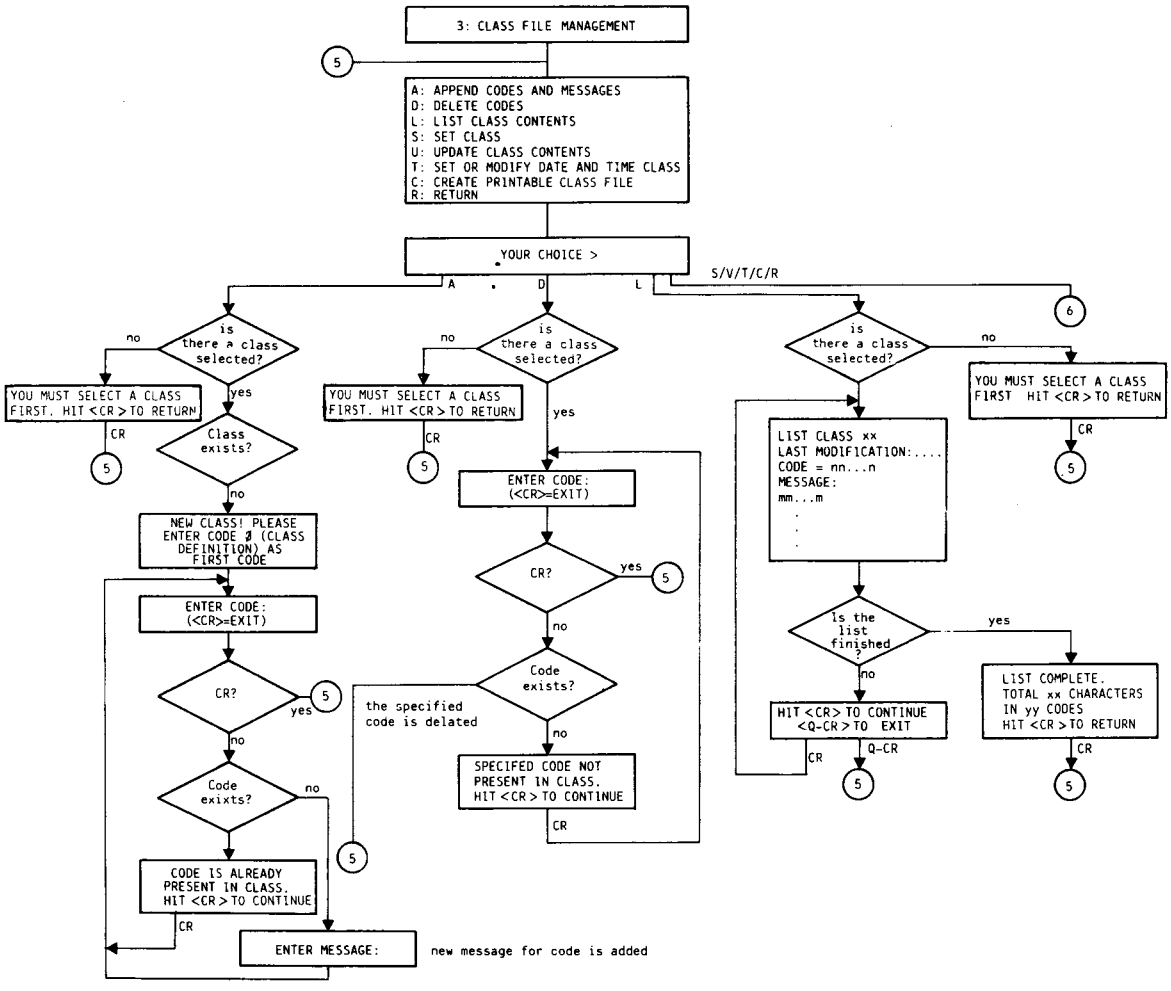


Fig. 4.MKCEM-5 MKCEM Command - Operator Interface (cont.)

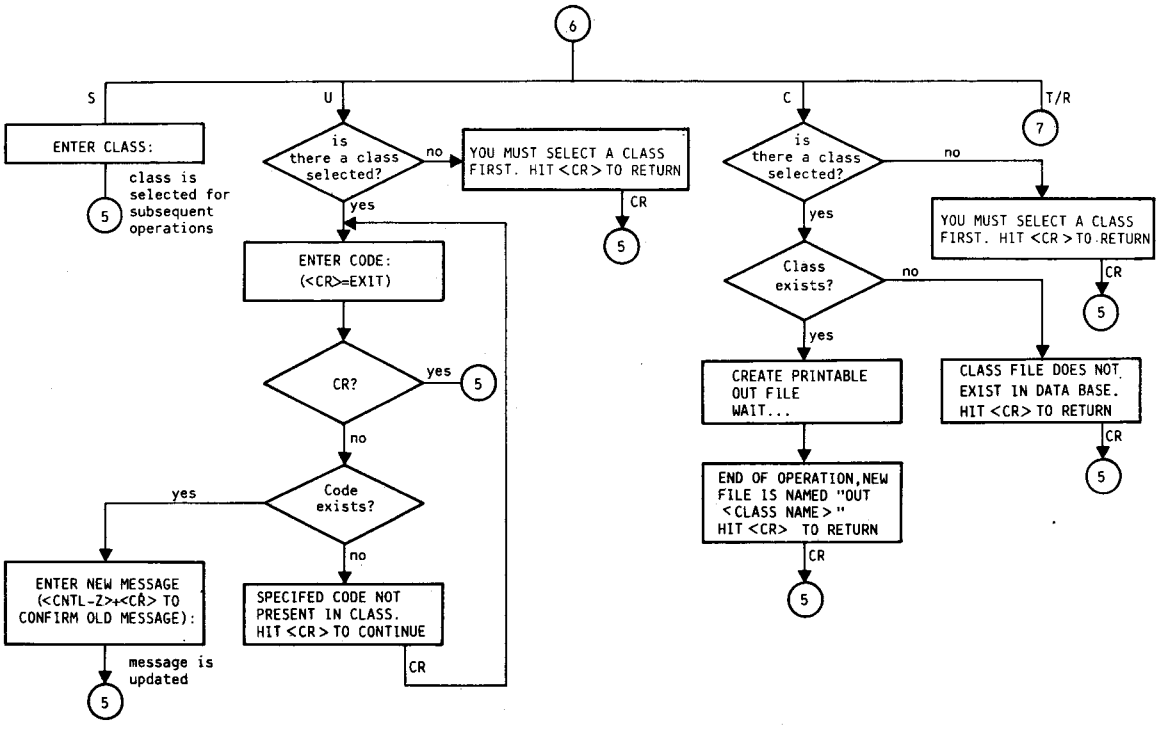


Fig. 4.MKCEM-6 MKCEM Command - Operator Interface

The third choice in the main menu allows the management of the classes containing the codes and messages with the following operations:

```
A: APPEND CODES AND MESSAGES
D: DELETE CODES
L: LIST CLASS CONTENTS
S: SET CLASS
U: UPDATE CLASS CONTENTS
T: SET OR MODIFY DATE AND TIME CLASS
C: CREATE PRINTABLE CLASS FILE
R: RETURN

YOUR CHOICE >
```

A	Adds codes and messages to a class.
D	Deletes a code and the corresponding message of the specified class.
L	Displays the messages in the specified class.
S	Creates or defines the class on which next commands are to be performed.
U	Modifies the message corresponding to the specified code.
T	Defines or modifies the date-related messages.
C	Creates a printable file of the specified class.
R	Returns to the main menu.

YOU MUST SELECT A CLASS FIRST. HIT <CR> TO RETURN

The class must be defined before another operation is performed on it. Press **CR** to return to the menu.

NEW CLASS! PLEASE ENTER CODE 0 (CLASS DEFINITION) AS FIRST CODE

A request has been made to add codes and messages to a class that has just been created. The first code to be created must be 0 (for the definition of the class).

ENTER CODE (<CR> = EXIT):

Prompts for the code to insert in the class.

CR Returns to the main menu.

CODE IS ALREADY PRESENT IN CLASS. HIT <CR> TO CONTINUE

The code to be added is already defined in the class. Press **CR** to return to the code prompt.

ENTER MESSAGE:

Prompts for the message related to the new code.

Note: The message may cover several lines. All the characters entered are considered, even **CR** (carriage return). The user may press **CONTROL A** to go to the beginning of the line, without the carriage return being included in the message. At the beginning of the last line of the message the user must enter **CONTROL Z** followed by **CR.**

SPECIFIED CODE NOT PRESENT IN CLASS. HIT <CR> TO CONTINUE

The code specified for deletion or update is not in the class. Press **CR** to return to the code prompt.

LIST CLASS xx LAST MODIFICATION : .....

CODE = nn...n
MESSAGE :
mm...m
.
.
.

When requesting the display of the contents of the class "xx", the information obtained is as follows:

- codes and corresponding messages
- date of last modification of the class file.

LIST COMPLETE. TOTAL xx CHARACTERS IN yy CODE(S). HIT <CR> TO RETURN

At the end of display, the total number of characters in the message ("xx") and the total number of codes ("yy") in the class is given. Press **CR** to return to the menu.

HIT <CR> TO CONTINUE, <Q - CR> TO EXIT

If the information covers more than one page, the user may press **CR** to display the next screen page, and **Q** followed by **CR** to interrupt the display and return to the menu.

ENTER CLASS:

Prompts for the class name to which the next requests will relate.

ENTER NEW MESSAGE (<CNTL-Z>+<CR> TO CONFIRM OLD MESSAGE)

Prompts for the new message to be entered. The user may type **CONTROL Z** followed by **CR** to confirm the existing message.

CLASS FILE DOES NOT EXIST IN DATA BASE. HIT <CR> TO RETURN

The class file specified for printing is not in the data base $CEM. Press **CR** to return to the menu.

CREATE PRINTABLE OUT FILE. WAIT...

The printable file for the specified class is created in the work directory.

END OF OPERATION, NEW FILE IS NAMED 'OUT<CLASS-NAME>'. HIT <CR> TO RETURN

Creation of the printable file has ended. The name of the new file is OUTCExxxx, where "xxxx" is the name of the class. Press **CR** to return to the menu.

Note: The file that has been created can be printed using the command LPR.

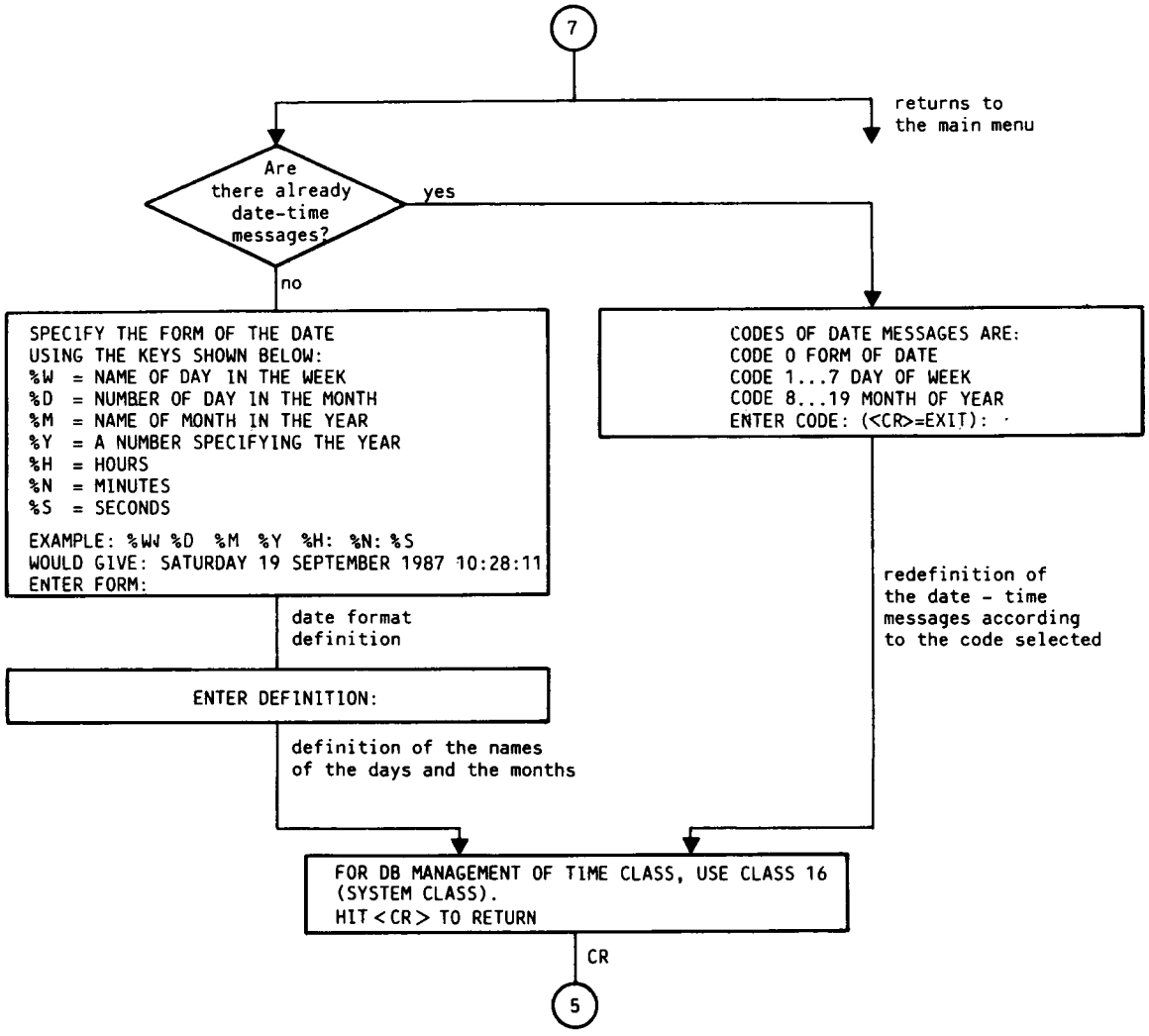


Fig. 4.MKCEM-7 MKCEM Command - Operator Interface

If the date-related messages are configured for the first time, the following screen page is displayed:

```
SPECIFY THE FORM OF THE DATE, USING THE KEYS SHOWN BELOW:

    %W = NAME OF DAY IN THE WEEK
    %D = NUMBER OF DAY IN THE MONTH
    %M = NAME OF MONTH IN THE YEAR
    %Y = A NUMBER SPECIFYING THE YEAR
    %H = HOURS
    %N = MINUTES
    %S = SECONDS

    EXAMPLE: %W %D %M %Y %H:%N:%S
    WOULD GIVE : SATURDAY 19 SEPTEMBER 1987 10:28:11

    ENTER FORM:
```

Definition of the date format is requested by means of the following keys:

%W represents the day of the week

%D represents the day in the month

%M represents the month

%Y represents the year

%H represents the time in hours

%N represents the time in minutes

%S represents the time in seconds.

An example of date format definition is also displayed.

ENTER DEFINITION:

This is a request for the definition of the names of the days of the week and the names of the months of the year.

If the date-related messages have been configured already, the format of the date, day of the week, and month of the year can be modified by means of the following menu, by selecting the appropriate code:

```
CODES OF DATE MESSAGES ARE:

CODE 0 FORM OF DATE
CODE 1...7 DAY OF WEEK
CODE 8...19 MONTH OF YEAR

ENTER CODE:
```

0	to redefine the date format
1..7	to specify the number of the day whose format is to be modified, e.g. 1 for Sunday, 2 for Tuesday, etc.
8..19	to specify the number of the month whose format is to be modified, e.g. 8 for January, 9 for February, etc.

```
FOR DB MANAGEMENT OF TIME CLASS, USE CLASS 16 (SYSTEM CLASS).
HIT <CR> TO RETURN
```

This message reminds the user that date and time related messages are contained in class 16 (the system class).

Example of date definition

If the date is required in the format

```
Mon Oct 27 14:00:10 1986
```

the user must:

- define the format for the date, as follows:

```
%W %M %D %H:%N:%S %Y
```

- specify the required names for all the days and months:

```
   .
   .
OCTOBER: Oct
   .
   .
MONDAY: Mon
```

ERROR IN CREATE FILE

Error during the creation of the printable file.

ERROR IN DISCONNECT FILE CLASS

Error during the release of the connection to the class file.

ERROR IN INPUT MESSAGE FUNCTION

Error while reading the message.

FAULT IN CONNECT TO /IPL/ETC, HIT <CR>

Error while connecting to the file /IPL/ETC. Press **CR** to return to the menu.

FAULT IN COPY DB, HIT <CR>

Error during the installation of the system data base. Press **CR** to return to the menu.

FAULT IN OPERATION: error message

Error while executing the utility. The "error message" specifies the nature of the error (see Appendix A).

INVALID PARAMETER(S)

The parameter specified in the command is not correct.

INVALID PATH NAME

The path name specified in the command is not correct.

OPERATION NOT ALLOWED, YOU CAN ONLY READ ON SPECIFIED PATH NAME. HIT <CR>

The operation requested is not allowed, as the user only has read access right on the specified directory. Press **CR** to return to the main menu.

NO REMOTE ACTION - INVALID OPERATION

It is not possible to operate on remote data bases.

PRIVILEGED OPERATION

The operation requested may only be executed by the system administrator (ROOT).

REMOVAL OF OLD DB NOT OK. <CR> TO CONTINUE

The removal operation of the old data base did not complete successfully. Press **CR** to return to the menu.

SECURITY VIOLATION ON CONNECT TO <path name>

The user does not have any access right to the directory specified in "path name".

THERE IS NO DATA BASE IN THIS DIRECTORY, HIT <CR>

There is no working data base in the directory specified in the command.

TOO MANY CLASSES IN DATA BASE, HIT <CR>

It is not possible to insert new classes in the data base, which may contain 64 classes at most.

TOO MANY CODES IN THIS CLASS! APPEND NOT POSSIBLE. HIT <CR>

It is not possible to insert new codes in this class. A class may contain at most 512 messages.

YOU CANNOT DELETE CODE 0, UPDATE ONLY! HIT <CR>

It is not possible to delete the message associated with code 0, which may only be updated. Press **CR** to return to the menu.

Initialises the first track (track 0) of a floppy disc. The command must always be executed before the MKVOL command for the creation of a main volume. It is only necessary for the floppy disc to be initialised to be loaded into the drive. If the floppy disc already contains a volume, the user is asked for confirmation before continuing.

where:

dev-name is the name of the drive on which the floppy disc resides, from FL1 to FL4.

"G is an optional parameter disabling the interactive mode. This allows the command to be used in Shell procedures.

OPERATOR INTERFACE

VOLUME DESCRIPTOR

```
VOLUME LABEL    : MCL
RELEASE NUMBER  : 0
RELEASE LEVEL   : 0
```

OK:

The floppy disc in the specified drive already contains a volume. The user should confirm whether or not to overwrite the data and reformat the floppy disc.

Y Reformats the floppy disc: a new environment is created and the previous data on the disc is lost.

N Aborts the MKENV operation: the data on the floppy disc is not overwritten.

The following diagram shows the format of track 0 after the command has been executed.

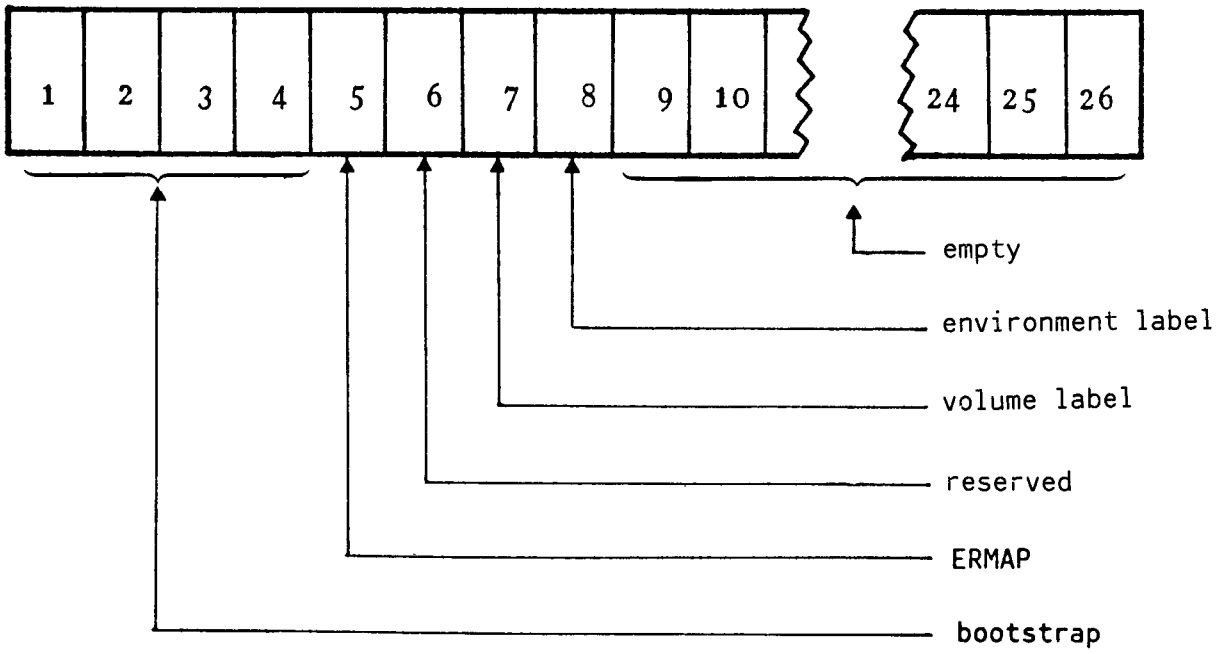


Fig. 4.MKENV-1 MKENV Format of Track 0

Examples

1. MKENV FL1

2. MKENV FL2 "G

Used for the reception and handling of all messages sent to the master workstation.

All the messages in the queue are read and the contents of each one is displayed in FIFO (First In First Out) order. When the last message has been displayed and the queue is empty, the operator is asked if he wants to wait for further messages or not.

The command also allows the operator to reply to a user who sent a message and is waiting for a response.

This command has no parameters.

Characteristics

1. When the operator is waiting for new messages, the terminal is reserved for the MTA command, and therefore not available. The only way to use the terminal is to kill the MTA command with the KILL command, which may be entered in two ways:

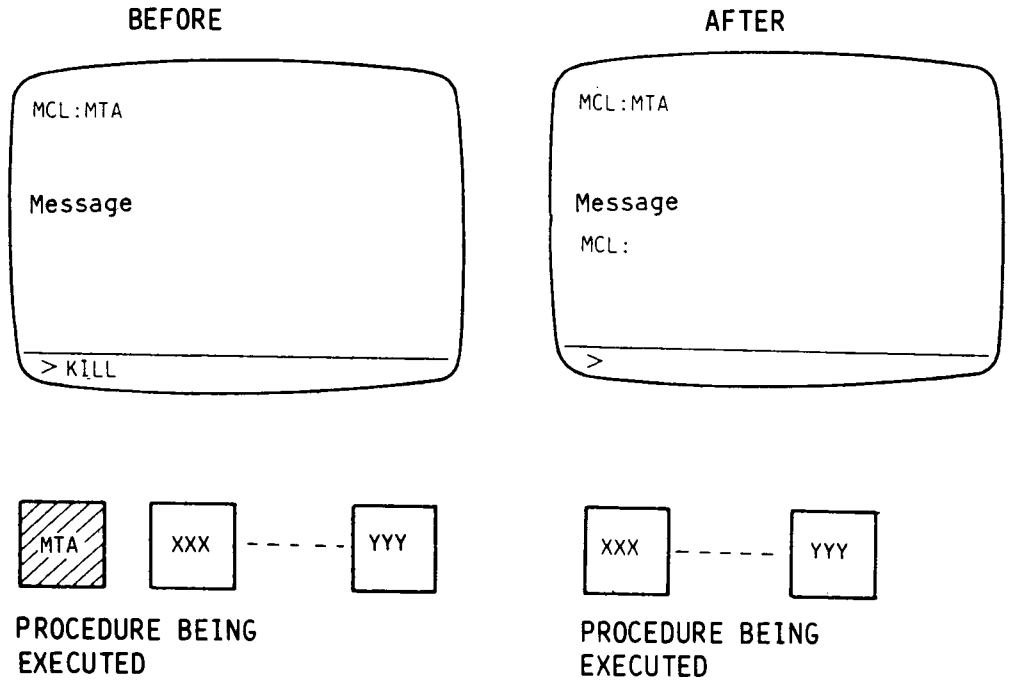
   - by pressing **CONTROL K**

   - by typing the command on the system line (which is still accessible to the user).

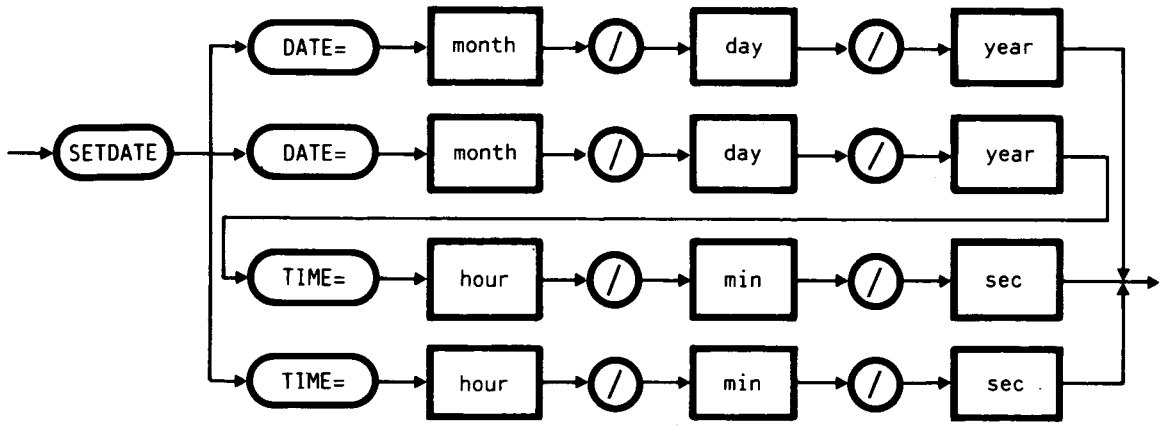
2. The message handling on the master workstation makes use of the buffers supplied by the port mechanism (see the manual "MOS Basic Architectural Concepts"). The size of the message queue depends therefore on the number of system buffers available.

Example

Reception of a message on a master terminal and killing of the MTA process so as to use the terminal for other processes.



Sets or modifies the system date or time.



where:

DATE specifies the date:

month is the month code, from JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC.

day is a two digit value in the range (01..31).

year is a two digit value in the range (00..99).

TIME specifies the time:

hour is a two digit value in the range (00..23).

min is a two digit value in the range (00..59).

sec is a two digit value in the range (00..59).

The parameters DATE and TIME are optional; if either is not specified, the previous value is retained.

Characteristics

The system checks the date entered for errors. For example:

FEB/30/83

will not be accepted by the system.

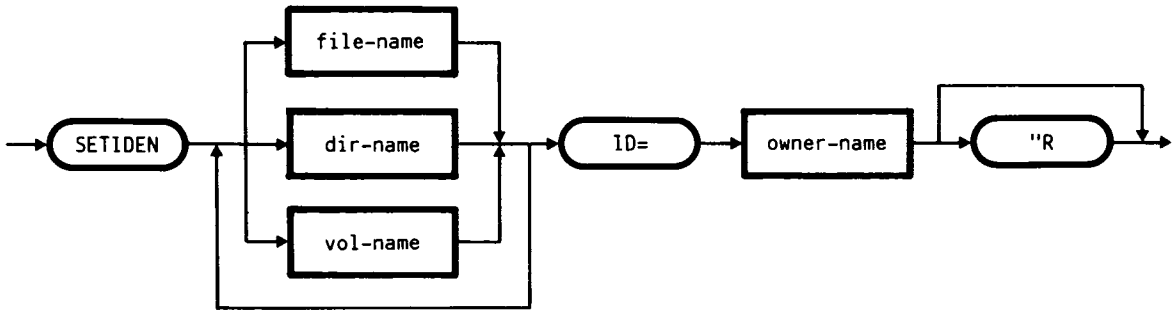
Examples

1. SETDATE DATE=MAR/27/82 TIME=15/36/12

2. SETDATE DATE=FEB/02/82

3. SETDATE TIME=12/15/00

This privileged command enables the owner of a file, a directory, or a volume to be changed.



where:

file-name is the name or the path name of the file.

dir-name is the name or the path name of the directory.

vol-name is the name or path name of the volume.

owner-name is the owner name to assign to the file, directory, or volume.

"R is an option which allows the user to change the owners of all the objects contained in a directory or volume, in recursive fashion.

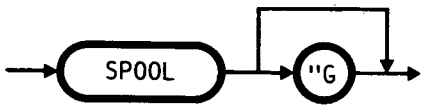
Characteristics

1. If the file name identifies a positional or keyed file, the command SETIDEN also changes the owner of all the associated files. If for example the owner of a keyed file is changed, the owner of the corresponding index file is also changed.

2. The name of the owner, specified in the parameter "ID", must correspond to a login name in the PASSWD file.

3. The recursive option **"R** is only significant if the SETIDEN command is used on a directory or a volume.

SPOOL gives the system administrator control of printing in the SPOOL environment, by means of all the functions described below.



where:

"G is the option enabling management of the global SPOOL system in a distributed configuration.

SPOOL FUNCTIONS

- Assign a class to an unspooler. As a class corresponds to a job queue, this is a means of assigning all the jobs in that queue to the particular printer controlled by that unspooler. Depending on how the printing set-up was configured, this may put the printer into "exclusive mode", only capable of printing jobs belonging to that queue (class) whilst this printing phase is active. Unless removed, this class-printer link is stored, and re-established at the next start-up.

- "Unlink" a class from an unspooler. The printer becomes available for direct printing only, and not for printing the SPOOL class; any jobs in that class will not be printed until the class is re-assigned.

- Start a class (the class is enabled for printing).

- Stop a class (print jobs in the class will not be printed, but new files can be added to the class (queue)).

- Hold any print job which has not started printing, preventing it from being printed.

- Ready (that is, release for printing) any print job which is in HOLD.

- Kill a print job.

- Change the priority of a print job.

- Suspend a print job in progress in a particular class, and therefore on a particular printer.

- Restart a suspended print job, from the beginning, from the stopping point, or from a point identified by a string.

- Show the spooling system environment in two screen pages, detailing classes, unspoolers, and existing links.

- Transfer one or more print jobs from one class to another. The attributes of the print job are not modified during this operation.

In common with other users, the system administrator can also:

- list a class (job queue for a particular printer and unspooler); this shows whether the class is active or suspended, existing connections, numbers of jobs enqueued together with their status, priority, number of copies required, and the first 18 characters of their title

- show the status of a job by giving its number (from READY, HOLD, RUN), its priority, owner, the number of copies required and the number of characters to be printed, and its title.

Characteristics

1. In a local network, the system administrator (ROOT) can control spooling for machines other than his own.

2. A class and unspooler specified in a command from ROOT must both reside on the same machine in the network.

The system administrator is presented with the following menu of all the functions available.

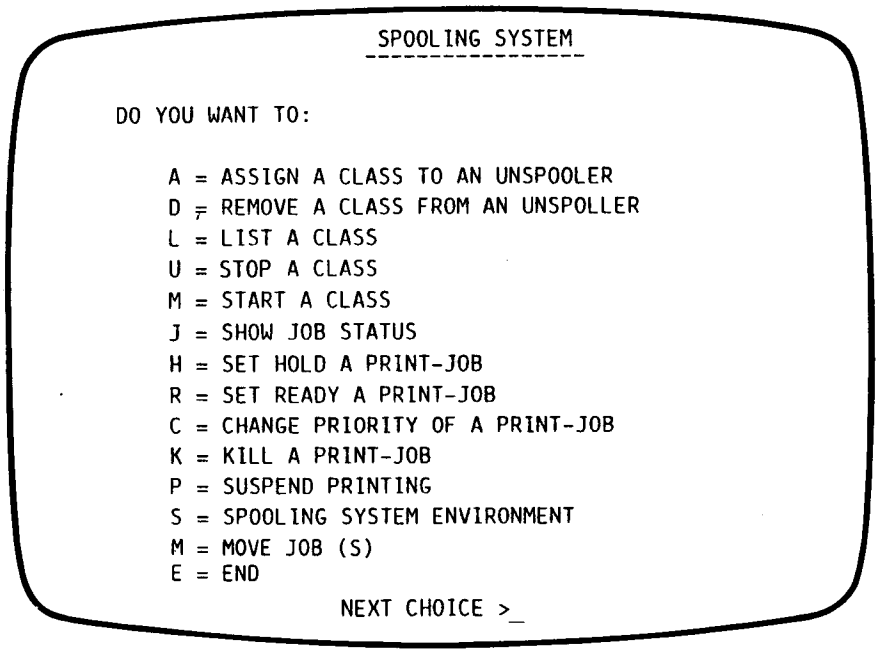


Fig. 4.SPOOL-1 SPOOL Command - System Administrator Menu

Select the function by entering the appropriate letter. The menu is then replaced by a request for a parameter, (see table and details below).

After a function has been performed, the user may perform another.

Pressing **.CR** will return the user to the main menu.

Alternatively, the user can trigger the function immediately by keying in the corresponding letter code, followed by the required parameters (separated by blanks); for instance:

for assigning the unspooler 2 to class 5, the user would type:

A 2 5

Parameters

SPOOL FUNCTION	COMMAND LETTER	PARAMETERS REQUIRED parameter1	parameter2	parameter3
assign a class to an unspooler	A	unspooler	class	-
remove a class from an unspooler	D	unspooler	class	-
list a class	L	class	-	-
stop a class	U	class	-	-
start a class	B	class	-	-
show job status	J	class	job number	-
set hold a print job	H	class	job number	-
set ready a print job	R	class	job number	-
change priority of a print job	C	class	job number	priority
kill a print job	K	class	job number	-
suspend printing	P	unspooler	-	-
show spooling system environment	S	-	-	-
move a print job	M	class	class	job number

Tab. 1 SPOOL Function Parameters

Valid parameter values are as follows:

- unspooler: a number from 1 to 16
- class: a number from 1 to 16
- job number: a number from 1 to 63, as allocated by SPOOL when the job is submitted
- priority: a number in the range 1 to 127. The job with the lowest priority number in a queue is printed first. Jobs are automatically allocated priority 32, which may be changed by the system administrator.

Note

If no job number is specified for the move function **M**, all the print jobs in the spooling class queue are transferred to the destination class, keeping their original attributes.

USERMAN allows the system administrator to register in the system the identities of authorised users, so that they can "login" when necessary; normal users may only use the listing facility.

The parameters are requested in interactive mode.

USERMAN FUNCTIONS:

- Create a group of users.
- Remove a group of users.
- Create individual users within specified groups.
- Delete individual users.
- List all users (in registration time order), whether they have defined a password or not, the groups to which they belong, and the machine on which they have been created.
- List all groups (in registration time order), their current user members and the machine on which they have been created.

Characteristics

1. Before using USERMAN, the "Login" data base must have been created using MKLOGIN.
2. Unless otherwise specified, new users are added to the /IPL/USR directory of the local machine.
3. All USERMAN functions are available to user "ROOT", but normal users can only list users or groups (S or L).
4. USERMAN works only in interactive mode.
5. All group and user names must be different from each other.
6. The user group of a user must be specified when creating this user.
7. The system administrator (ROOT) is the first and only user in a newly installed system. As a user it can neither be removed nor recreated.

8. Users and user groups can only be removed from the machine on which they have been created.

OPERATION

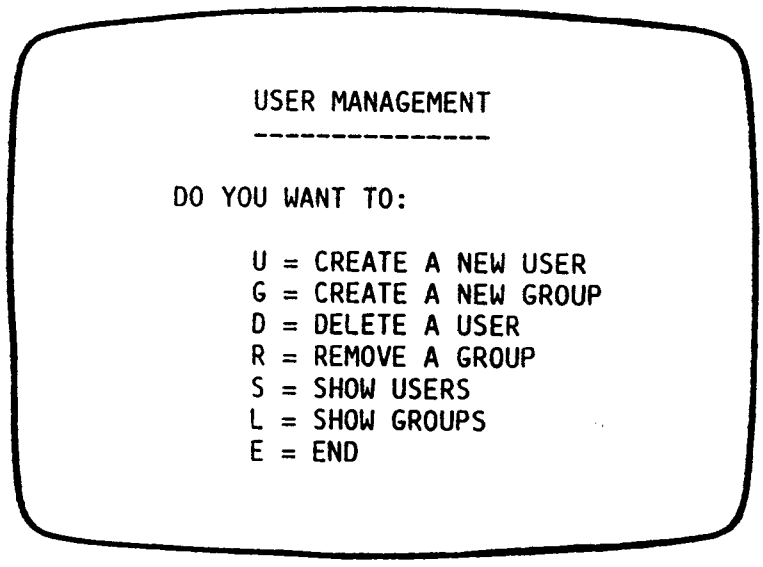


Fig. 4.USERMAN-1 USERMAN Command - System Administrator Menu

1. To select a function the user must enter the appropriate letter and then press **CR**.

2. Guiding messages follow prompting for parameter values. After each input press **CR**.

3. If an incorrect value is entered for a parameter, the function aborts and returns to the main menu. Invalid functions usually cause an error message to be displayed (see further on); this will stay on the screen until **CR** is pressed, then control returns to the main menu.

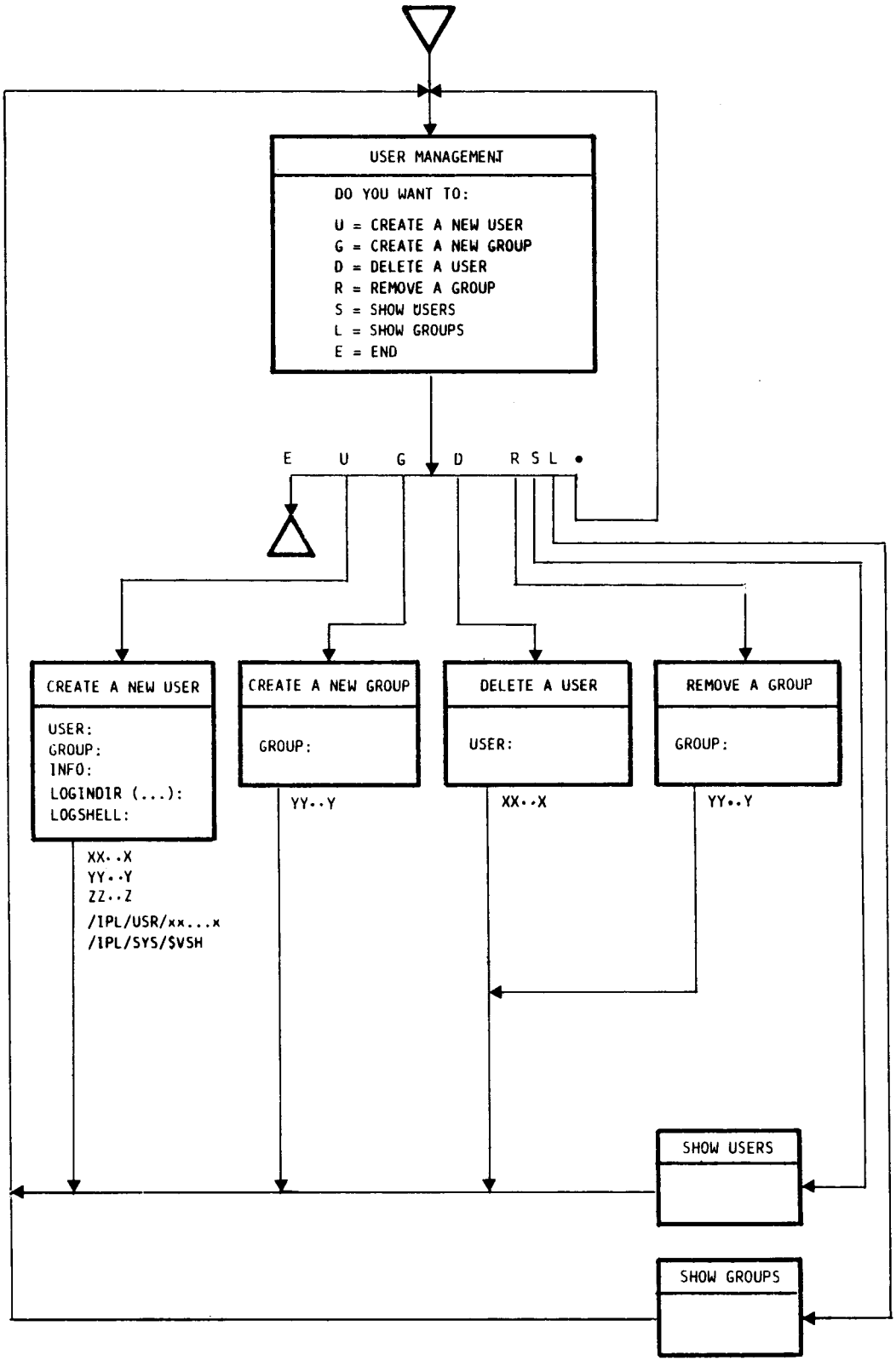


Fig. 4.USERMAN-2 USERMAN Command - Operator Interface

Parameters

Parameters required by USERMAN functions are shown below.

USERMAN FUNCTION	COMMAND	PARAMETER REQUIRED				
		param.1	param.2	param.3	param.4	param.5
Create new user	U	user	group	comment	home dir.	Shell file
Create new group	G	group	-	-	-	-
Delete user	D	user	-	-	-	-
Remove group	R	group	-	-	-	-
List all users	S	-	-	-	-	-
List all groups	L	-	-	-	-	-

Tab. 1 USERMAN Functions Parameters

Values for the above parameters are described below:

USER:	Enter the name by which the user is known to the system, (maximum of 8 chararacters, must not include "/"), to be created or removed depending on the function.
.	Returns to the main menu.
CR	Entered on its own, proceeds to the next prompt.

GROUP:	Enter the group name (maximum of 8 characters, must not include "/"), to be created, added, or removed, depending on the function.
.	Returns to the main menu without doing anything.
CR	Entered on its own, proceeds to the next prompt.

INFO:	Optional field of 40 characters at most, irrelevant to the function. This may be any comment which the system administrator finds useful to record about the user: for instance "this user works on the system from 8.30am to 3.30pm".
CR	Entered on its own, proceeds to the next prompt.

LOGINDIR (DEFAULT IS /IPL/USR/<user>): /../NxMy

Give the path name for the user home directory if different from the default.

CR Completes the entry, or selects the default if entered alone; then goes on to the next prompt.

LOG SHELL:

Enter the path name of the Shell file, for example, /IPL/SYS/$VSH.

. Returns to the main menu without doing anything.

CR Completes the entry, and returns to the main menu.

Listing functions S and L

These require no parameters, and their operation is as follows:

S produces the display of users shown below, which remains on screen until removed by a **CR.**

Users are listed in "historical" order, that is, the user registered earliest on the system is first on the list.

USER	PASSWORD	GROUP	INFORMATION	MACHINE NAME
User A	Y	Group 1	(information	NxMy
User B	Y	Group 4	on the users)	NxMy
User C	Y	Group 2		NxMy
User D	N	Group 1		NxMy

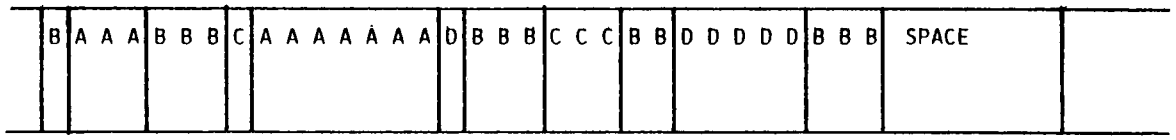
HIT <CR>

L lists groups and related information as shown below, also in "historical" order. The display remains on screen until removed by a **CR.**

```
GROUP          MACHINE NAME        MEMBERS
-------------------------------------------------------------------

Group 1            NxMy            User A User C User M
Group 2            NxMy            User D User S
Group 3            NxMy            User V User K User J User X

HIT <CR>
-------------------------------------------------------------------
```

VCOMPACT is used on hard discs volumes only (18/60/120/140 Mbyte), to unify volume fragments in order to optimise access times. VCOMPACT is guaranteed to function on volumes of up to 10000 extents, and may handle larger volumes.

Before VCOMPACT, file extents may be scattered thus:



Following VCOMPACT, fragments of files are collected so that all the extents of each file are stored contiguously, to appear thus:

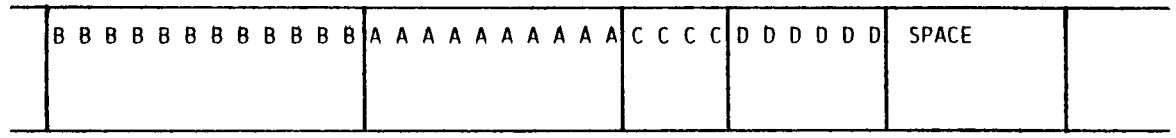


Fig. 4.VCOMPACT-1 Effect of VCOMPACT

VCOMPACT FUNCTIONS

VCOMPACT can:

- merge file system objects which are fragmented and scattered on disc, concentrating each into a single extent
- compress all file system objects towards the beginning of the volume, leaving a single, vacant space behind them for future use
- operate on a volume in read-only mode, to determine the actions that will be taken. Potential problems may thus be recognised before actually operating on a volume
- reduce output so that only error messages are reported to the user.

Characteristics

1. VCOMPACT is only available to the system administrator (ROOT).

2. VCOMPACT requires a dedicated system.

3. Parameters must be entered at the same time as the command.

4. VCOMPACT does not recover logically deleted space, or modify file allocation units.

5. Unlike most commands, the operation of VCOMPACT is more complete when no options are specified, because its defaults are optimised.

6. The file structures are not modified during the operation; they are updated only when compacting is complete.

7. Volumes to be operated on must be mounted. VCOMPACT cannot be used on other system objects.

Warnings

- Only kill a VCOMPACT operation in emergency: this will logically lose all the data on the volume.

- VCOMPACT can be such an extensive operation that it must never be run without full preparation (see "Operation" below).

- Ensure, before starting VCOMPACT, that none of the volumes needed contain files currently accessed by other processes.

- VCOMPACT cannot compact a volume currently being accessed by another process, therefore it cannot compact the volume containing the current working directory because this is already being accessed.

- Where the environment uses linked hard discs, the use of sub-volumes under /IPL enables VCOMPACT to be used on them without the inherent dangers.

OPERATION

The procedure to follow to use VCOMPACT is:

1. Dedicate the system: that is, ask all other users to exit from the system as soon as possible.

2. Check the volume. The volume intended for VCOMPACT must be free of defects, so check it using DISKCHECK (see MOS System Software Maintenance Tools User Guide.)

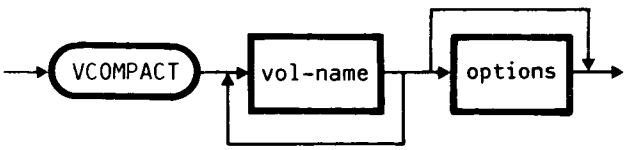
3. If anything is missing (PDD, extent, etc.), this will be indicated. In such a case, run VOLGC (see MOS System Software Maintenance Tools User Guide), which will correct any inconsistencies, but does not display the actual corrections.

4. Check the volume again. Run DISKCHECK to see the post VOLGC state of the file.

5. If required, run VCOMPACT in "read-only" mode.

6. Run VCOMPACT.

7. Only if VCOMPACT was run on /IPL, exit from MOS with LOGOUT and SHUTDOWN.



---

where:

vol-name is the name of the volume(s) to be compacted. Names must be separated by single spaces, and may be complete path names, or local names relative to the current working directory.

options selects a function:

C This compresses file fragments into a single, contiguous extent at the beginning of the disc, regrouping all the free space at the end of the volume. This option will not allow a simultaneous merge (**M**).

M This merges file system objects with multiple extents. This option does not allow a simultaneous compression (**C**), although some compression of the volume may occur during file merging.

R Read-only mode: simulates the compacting process without actually writing on the disc, anticipating any problems and showing the amount of disc I/O necessary to compact the volume.

S Silent mode: only error messages are reported. This is useful if VCOMPACT is executed from within a Shell procedure, or in batch mode.

If no options are specified, or **R** alone, the volume will be both compressed and merged.

Do not use **C** and **M** together, and use **S** only alone.

DEVICE NOT READY

Device disconnected or off, floppy disc not in drive or inserted wrongly, or drive flap open.

ERROR IN BITMAP for block @ nn

The blocks represented in the bit map are inconsistent with the structure of the volume. The error is in block nn.

ERROR IN OPTION

An error occurred while reading the options.

EXTENT TABLE SPACE EXCEEDED

The number of extents contained in the volume is greater than the maximum number (51870) that can be inserted in the extent tables of the utility.

HARDWARE ERROR

Hardware fault; for instance, a floppy disc is damaged or inserted badly in the drive.

INVALID EXTENT ADDRESS FOR PDD nn
OFFSET = nn
LENGTH = nn
ADDRESS = nn

While checking the PDDs of the volume, an error has been found on an extent of the PDD nn, due to an inconsistency on the volume data structure.

NO REMOTE ACTION - INVALID OPERATION

The remote operation requested is invalid, as the system is not distributed.

OPTION a IGNORED

An incorrect command option has been specified.

OVERLAPPING EXTENTS PDD nn
OFFSET = nn
LENGTH = nn
ADDRESS = nn

PDD nn
OFFSET = nn
LENGTH = nn
ADDRESS = nn

While checking the PDDs of the volume, two references to the same extent have been found. This indicates the presence of inconsistencies in the data structures.

SECURITY VIOLATION

Security violation of a file system item.

VOLUME SIZE EXCEEDED

An inconsistency in the volume data structure has been found while analysing the PDD table and building its memory image.

WARNING: THE VOLUME MAY BE CORRUPTED

The operation of the utility has been stopped by a KILL or a memory error. VCOMPACT terminates but the state of the volume data may be inconstistent.

Examples

1. VCOMPACT /VOL

   Compresses and merges the volume named /VOL.

2. VCOMPACT /VOL "R

   Simulates the compression and merge in read-only mode of the volume named /VOL. This involves going through the process of compressing and merging without actually rewriting the volume, but shows what results can be expected if it was actually done.

3. VCOMPACT /VOL1 /VOL2 "CR

   Compresses (does not merge) in read-only mode the volumes named /VOL1 and /VOL2, that is, show what would happen if the volume were to be compressed.

RESULTS

The VCOMPACT display shows the state of the volume(s) before and after compaction. If performed in read-only mode, the same display is produced to show what would have happened if extents had really been moved.

Display Example

The display following a VCOMPACT operation on /IPL/DR2, in read-only mode, is shown below:

The procedure checks the volume and shows what it finds:

```
MCL: VCOMPACT /IPL/DR2 "R

CONNECTING TO /IPL/DR2
CHECKING TYPE OF /IPL/DR2
OPENING /IPL/DR2
READING VOLUME DESCRIPTOR
```

The procedure summarises the contents of the volume:

```
VOLUME: /IPL/DR2

512 BYTES PER BLOCK
984576 BYTES IN VOLUME
PDD TABLE AT OFFSET 512
104 PDDS
208 EXTENTS
```

The procedure details the movement of each PDD (Permanent Dataset Descriptor) it has sorted and merged, and gives location details:

```
SORT: EXTENTS BY FILE SYSTEM OBJECTID AND OFFSET
MOVING
PDD# 4 OFFSET=  512    LENGTH=   512   ADDRESS=346012
COPYING SOURCE  346112 DESTINATION 374336 LENGTH  512
READING ADDRESS 346112 MEMORY OFFSET    0 LENGTH  512
WRITING MEMORY OFFSET 0 ADDRESS    974336 LENGTH  512
PDD# 4  OFFSET=   512  LENGTH=   512   ADDRESS=974336
END MOVE

SORTING EXTENTS BY ADDRESS
MERGING
PDD#  4 OFFSET=     0   LENGTH=   512   ADDRESS=973824
PDD#  4 OFFSET=   512   LENGTH=   512   ADDRESS=974336
PDD#  4 OFFSET=     0   LENGTH=         ADDRESS=973824
END MERGE
```

Finally, the operations performed within this VCOMPACT are summarised as follows:

```
STATISTICS
629760 BYTES READ
629760 BYTES WRITTEN
10 ADDRESS SORTS WITH 29709 COMPARISONS
6 FILE SYSTEM OBJECT SORTS WITH 17621 COMPARISONS
CLOSING /IPL/DR2
DISCONNECTING /IPL/DR2
MCL:
```

APPENDICES

PART 4 - APPENDICES

INTRODUCTION TO PART 4

Part 4 contains all the appendices:

- Appendix A is a list of all system-produced error messages, their meaning, and action required.
- Appendix B details the limits of hardware and software objects significant in the Shell environment.
- Appendix C provides detailed and background information about magnetic tapes and their use.
- Appendix D provides detailed and background information about floppy discs and their use.

A. ERROR MESSAGES

This appendix contains a list of the following types of centralised error messages displayed by the Shell environment:

- Error messages issued by the operating system (class 16)
- BATCH error messages (class 2817)
- SPOOL error messages (class 2818)
- Error messages issued by other commands (class 2816)

These messages can be translated into other foreign languages as appropriate, and then stored into the $CEM message data base, using the command MKCEM described in this manual.

They have the following layout:

```
cc...c  (nn)  -  mm...m
```

where:

cc...c is the name of the error class

nn is the numeric code of the error (preceded by a * for the system errors)

mm...m is the explanatory message associated with the error.

MOS-CMD (*256) - TERMINAL DOWN

The terminal is off.

MOS-CMD (*257) - DIAGNOSTIC ERROR

Diagnostic error due to a peripheral hardware failure.

MOS-CMD (*258) - RECEIVE ERROR

Character reception error. An error occurred on the line connecting the peripheral.

MOS-CMD (*259) - HARDWARE ERROR

Hardware fault: for instance, a floppy disc is damaged or inserted badly in the drive.

MOS-CMD (*260) - SYSTEM ERROR

Software fault: inconsistency in the system data areas.

MOS-CMD (*261) - TABLE OVERFLOW

Overflow in one of the system tables.

MOS-CMD (*262) - OUT OF DISK SPACE

There is no more space on the disc.

MOS-CMD (*263) - FATAL ERROR

Inconsistency in the internal system tables.

MOS-CMD (*264) - READ ONLY SEGMENT

An attempt has been made to violate the read-only protection on a segment by trying to write to it.

MOS-CMD (*265) - SEGMENT VIOLATION

The user has attempted to access a system segment.

MOS-CMD (*266) - SEGMENT LENGTH FAULT

An attempt has been made to access a segment beyond its actual boundaries.

MOS-CMD (*267) - UNALLOCATED SEGMENT

An attempt has been made to access an unallocated segment.

MOS-CMD (*268) - BAD SEGMENT HANDLING

An attempt has been made to read or write a segment containing executable-only code.

MOS-CMD (*269) - INSTRUCTION ERROR

An attempt has been made to execute a non-existing instruction.

MOS-CMD (*270) - PRIVILEGED INSTRUCTION

An attempt has been made to use a privileged instruction, while the processor is in normal state.

MOS-CMD (*271) - SEGMENT VIOLATION

Self-explanatory.

MOS-CMD (*272) - NO PROCEDURE

A procedure which is not available on the system has been called.

MOS-CMD (*273) - NO ROUTING

In a distributed system, the machine containing the requested resource (file, directory, or volume) is not available for one of the following reasons:

- the machine is not connected to the network
- the machine has been identified wrongly
- the machine is faulty.

MOS-CMD (*274) - RESULT DOUBTFUL

An operation on a remote resource has been requested in a distributed system, but no news have been received about its execution. The user is advised to restart the operation.

MOS-CMD (*275) - OPERATION ABORTED

During a remote command request in a distributed system, a message transmission error occurred.

MOS-CMD (*512) - RECORD TOO LONG

The data to transfer is too long for the space available.

MOS-CMD (*513) - INVALID OPERATION

Message displayed in various circumstances, when an operation is invalid. For example, this message appears in the following situations:

- an invalid conversion has been specified in the CONVERT command
- a user has attempted to start a command reserved for the system administrator
- a reference has been made to a byte-stream file in a user program as if it was another type of file, or the write mode parameter has been specified inconsistently.

MOS-CMD (*514) - INVALID ID

The file identifier is invalid.

MOS-CMD (*515) - INVALID PARAMETERS

An incorrect parameter has been specified, or a parameter points to an area that is either inaccessible or cannot be written to.

MOS-CMD (*516) - ERROR IN THE STRING

The file on which a write request was made contains one or more strings without end characters, such as line feed or carriage return.

MOS-CMD (*517) - SECURITY VIOLATION

Security violation of a file system item.

MOS-CMD (*518) - NOT YET IMPLEMENTED

The operation requested is incompatible with the current implementation.

MOS-CMD (*519) - INCORRECT SCOPE

The output variable is not local to the caller.

MOS-CMD (*520) - INCORRECT STATE

The state of a process or family is not consistent with the operation requested on it.

MOS-CMD (*521) - ILLEGAL INDEX

Array index out of boundaries, or index outside the segment table.

MOS-CMD (*522) - CHANNEL DESTROY

A family channel has been destroyed.

MOS-CMD (*523) - CHANNEL BUSY

Read attempt on busy channel.

MOS-CMD (*526) - NO WRITERS

Read suspension on a channel on which no-one can write.

MOS-CMD (*768) - MEMORY CONFLICT

Load operation on segments already allocated.

MOS-CMD (*769) - BAD FILE IDENTIFIER

Incorrect file identifier.

MOS-CMD (*770) - TYPE NOT COMPATIBLE

Conflict between type and attribute of a segment being loaded.

MOS-CMD (*771) - INVALID FORMAT

Attempt to load an l_module in the wrong format.

MOS-CMD (*772) - ILLEGAL SEGMENT LENGTH

Self-explanatory.

MOS-CMD (*773) - INVALID SEGMENT NUMBER

Self-explanatory.

MOS-CMD (*774) - PROCESS INTERRUPT

Process interrupted by an external event.

MOS-CMD (*775) - NO HALT

The process is terminated without a HALT being performed.

MOS-CMD (*776) - PROCESS DESTROY

Signals the destruction of a process.

MOS-CMD (*778) - PROCESS SUSPEND

Signals the suspension of a process.

MOS-CMD (*779) - BREAK POINT

A process has reached a breakpoint.

MOS-CMD (*780) - NOT EMPTY AREA

The address space is not empty.

MOS-CMD (*1024) - MEMORY OVERFLOW

Not enough memory to create a new segment, expand an already existing segment, load or start a program, or create a new family.

MOS-CMD (*1025) - TOO MANY PROCESSES

There are too many processes running simultaneously.

MOS-CMD (*1026) - ARRAY TOO LONG

The size of the array obtained exceeds the predefined boundaries.

MOS-CMD (*1280) - BYTESTREAM OPERATION

Operation of type positional or keyed attempted on a byte-stream file.

MOS-CMD (*1281) - POSITIONAL OPERATION

Operation of type keyed attempted on a positional file.

MOS-CMD (*1282) - INCONSISTENT DATABASE

Index inconsistent with the contents of the file.

MOS-CMD (*1283) - NO PRIMARY INDEX

The primary index has been deleted.

MOS-CMD (*1284) - DUPLICATED KEY

There is a duplicate secondary key.

MOS-CMD (*1285) - INCOMPLETE RECORD

An incomplete record has been read.

MOS-CMD (*1536) - TIMEOUT

The resource access timeout has expired.

MOS-CMD (*1537) - RECORD LOCKED

A record is inaccessible because it has already been locked by another process.

MOS-CMD (*1538) - DEVICE NOT READY

Device disconnected or off, floppy disc not in drive or inserted wrongly, or drive flap open.

MOS-CMD (*1539) - INVALID PATHNAME

The path name is invalid.

MOS-CMD (*1540) - NAME ALREADY EXISTS

The name of the file, directory, or volume exists already.

MOS-CMD (*1541) - NAME NOT FOUND

File, directory, or volume name not found.

MOS-CMD (*1542) - NOT EMPTY DIRECTORY

The directory is not empty and cannot be deleted.

MOS-CMD (*1543) - END OF FILE

Attempt to read beyond the end of the file.

MOS-CMD (*1544) - OUT OF BOUNDS

Attempt to read a segment using an offset which is out of the segment's boundaries.

MOS-CMD (*1545) - RECORD NOT FOUND

There is no valid record at the position specified.

MOS-CMD (*1546) - RECORD ALREADY EXISTS

The record to insert exists already.

MOS-CMD (*1547) - KEY NOT FOUND

The key specified has not been found in the index, therefore the keyed file cannot be accessed.

MOS-CMD (*1548) - KEY ALREADY EXISTS

Attempt to create a duplicate key.

MOS-CMD (*1549) - OPERATOR INTERVENTION

The operator has modified the working conditions whilst work was taking place: for instance, a floppy disc has been replaced whilst it was being used.

MOS-CMD (*1550) - ALIASED NOT FOUND

The aliased file does not exist.

MOS-CMD (*1792) - END OF PAGE

End of page or document during printing.

MOS-CMD (*1793) - BAD PAPER POSITION

The document is not inserted correctly in the printer front feed, or the printer is in local.

MOS-CMD (*1794) - OPERATOR REQUEST

Operator intervention is requested on the printer or cash adapter.

MOS-CMD (*1795) - NO CHARACTERS

There are no characters waiting in the circular buffer of the terminal keyboard or the PIN pad.

MOS-CMD (*1796) - BREAK OCCURRED

A breakpoint has been requested using the corresponding key, a software interrupt has been sent, or a timeout has expired.

MOS-CMD (*1797) - TERMINAL ON

The specified terminal is on.

MOS-CMD (*1798) - PARITY WARNING

Parity error on the specified key.

MOS-CMD (*1799) - WEAK KEY WARNING

The key specified belongs to the set of weak keys (see the ENCRYPTION Driver Interface Programmer Guide).

MOS-CMD (*1800) - TOO MANY PIN-CHECK ATTEMPTS

The requested check function has exceeded the threshold of warnings related to the number of unsuccessful comparisons made.

MOS-CMD (*2048) - UNIT DOWN

The peripheral unit is down.

MOS-CMD (*2049) - UNIT NOT AVAILABLE

Attempt to use a peripheral which is not defined in the configuration file of the workstation.

MOS-CMD (*2050) - RESOURCE BUSY

Shared resource in use by another process.

MOS-CMD (*2051) - LINE ERROR

The peripheral has detected a transmission error on the line.

MOS-CMD (*2052) - BADLY INSERTED DOCUMENT

A document is jammed in the front feed of the printer or cheque reader/writer, or the cash adapter is jammed.

MOS-CMD (*2053) - NO DOCUMENT

There is no document inserted in the front feed of the printer, the timeout expired before the insertion of a badge or cheque in the badge or cheque reader/writer, or there is a read/write attempt on a document with no magnetic strip.

MOS-CMD (*2054) - CARTER IS OPEN

The top of the printer is open.

MOS-CMD (*2055) - LOCAL

The printer is in local.

MOS-CMD (*2056) - VERIFY OR READ ERROR

Attempt to read or write on a magnetic strip.

MOS-CMD (*2057) - VIRGIN STRIP

The magnetic strip of a document or badge does not contain any data.

MOS-CMD (*2058) - PRINT HEAD BLOCKED

Self-explanatory.

MOS-CMD (*2059) - END OF INPUT

The operator has pressed **CONTROL D** (end of file) as the first character during an open session on the terminal.

MOS-CMD (*2060) - BADGE INSERTED

There is a badge in the badge reader/writer already.

MOS-CMD (*2061) - END OF TAPE

The ribbon of the daisy printer or cheque reader/writer has finished.

MOS-CMD (*2062) - OFF OCCURRED

Signals the occurence of an ON-OFF-ON transition of the terminal, in the case where the driver calls did not succeed during the OFF phase, and therefore the user did not receive the 'TERMINAL DOWN' signal.

MOS-CMD (*2063) - PRINTER TIMEOUT

The timeout set for a printer reply has expired.

MOS-CMD (*2064) - DISABLED FUNCTION

The check function requested has been disabled after reaching the threshold of warnings related to the number of unsuccessful comparisons made.

MOS-CMD (*2065) - OVERFLOW NUMBER GETTING PASSWORD OF ENCRYPTION BOARD

The threshold value of errors in password checking has been reached.

MOS-CMD (*2066) - PIN NOT MATCHED

The PIN entered is incorrect.

MOS-CMD (*2067) - WRONG PASSWORD

The password entered is wrong.

MOS-CMD (*32767) - MESSAGE NOT AVAILABLE

The expected message is not configured in the error class contained in $CEM.

BATCH (19) - ACCESS NOT ALLOWED

The user has tried to modify the details of another user's job.

BATCH (20) - WARNING: BATCH ENVIRONMENT NOT AVAILABLE

The batch environment has not been configured.

SPOOL (14) - WARNING: SPOOLING SYSTEM NOT AVAILABLE

Self-explanatory.

SPOOL (24) - ACCESS NOT ALLOWED

In the SPOOL command, a function reserved to the system administrator has been requested, or an attempt to delete or modify another user's print job has been made.

SPOOL (25) - CANNOT ACCESS CONFIGURATION FILE

The command cannot be performed because the data in the configuration file cannot be accessed.

SPOOL (52) - AT MOST 99 COPIES ALLOWED

When using the LPR command, no more than 99 copies can be printed for each print job.

SPOOL (55) - PRINTER NOT CONNECTED

The printer is not connected to the system properly.

SPOOL (56) - ERROR IN OPENING PRINTER

An error occurred during an open printer operation.

SPOOL (57) - INVALID FILE TYPE

The type of the file specified in the command is not suitable for printing.

SPOOL (58) - ERROR IN PARAMETER(S)

A command parameter is either missing or specified wrongly.

SPOOL (60) - INVALID OPTION(S)

The specified option is incorrect.

SPOOL (62) - INVALID NUMERIC STRING

A numeric parameter has been given a character value.

SPOOL (63) - ERROR IN CONNECT ROOT --> ABORT

The job cannot be printed due to an error during the connection to the ROOT machine.

SPOOL (64) - WARNING: CLASS NOT AVAILABLE

The SPOOL class requested is not available.

SPOOL (68) - PRINTER OF WORKSTATION SYSPRTx NOT AVAILABLE

The workstation printer "x" is not available.

SPOOL (70) - ** TERMINAL WAS DOWN **

After a print job request, the terminal was turned off. This message appears when the terminal is turned on again.

SPOOL (77) - UNABLE TO PRINT JOB nn

This message is output on the printer when a print file request is made with the option "no copy" and the file is already in use.

SPOOL (78) - ** JOB KILLED **

This message is output on the printer following the deletion request of a job currently printing.

SPOOL (79) - PRINTER SYSPRTx NOT AVAILABLE

Self-explanatory.

SPOOL (98) - INVALID PATH_NAME

The path name specified in the command is invalid.

SPOOL (124) - PRINTER SYSPRTx DOWN

Self-explanatory.

SPOOL (125) - PRINTER SYSPRTx IN LOCAL

Self-explanatory.

SPOOL (126) - CARTER IS OPEN ON PRINTER SYSPRTx

The top of the printer "x" is open.

MOS-CMD (100) - ILLEGAL OPTION(S)

One or more of the options specified in the command are incorrect.

MOS-CMD (101) - ILLEGAL KEY PARAMETER

The keyed parameter specified in the command is incorrect.

MOS-CMD (102) - INVALID PARAMETER(S)

One or more commands have been declared wrongly (syntax or contents).

MOS-CMD (103) - <WARN> VALUE TOO LONG

The string defined with the keyword WARN is too long.

MOS-CMD (104) - MISSING <TIMEOUT> PARAMETER

There is no key parameter TIMEOUT in the command syntax.

MOS-CMD (105) - INVALID OPERATION

The operation requested is invalid.

MOS-CMD (106) - INVALID OPTION

The option specified in the command is not allowed.

MOS-CMD (107) - ERROR IN PARAMETER(S)

One or more parameters have not been specified in the command syntax, or have been specified wrongly.

MOS-CMD (108) - WRONG DESTINATION

The destination parameter specified in the command is a file system item incompatible with the specified source file or directory.

MOS-CMD (109) - INVALID NUMERIC STRING

A non-numeric character has been entered in the command parameter.

MOS-CMD (110) - ERROR IN GETTING GROUP

An error occurred when reading the information file for the group specified in the FINGER command.

MOS-CMD (111) - INVALID NUMERIC STRING

A non-numeric character has been entered in the command parameter.

MOS-CMD (112) - INVALID KEY

The keyword KEY, used to create a keyed file, has been used when trying to create a byte-stream file.

MOS-CMD (113) - EXTERNAL SECONDARY KEYS ARE NOT ALLOWED

A secondary index cannot contain external record keys.

MOS-CMD (114) - INVALID INDEX TYPE

When creating a file index, the character used to specify the type of the secondary index is invalid.

MOS-CMD (115) - WRONG DISK

The main volume specified in the command is different from the floppy disc descriptor.

MOS-CMD (116) - INVALID DEV NAME

The device name specified in the command is incorrect.

MOS-CMD (117) - INVALID DIRECTORY

The directory path name specified in the command is invalid.

MOS-CMD (118) - INVALID DATE VALUE LENGTH

The length of the date value given is incorrect.

MOS-CMD (119) - INVALID TIME VALUE LENGTH

The length of the time value given is incorrect.

MOS-CMD (120) - BAD PASSWORD

The two instances of the new password defined in the command SETPASSW are different.

MOS-CMD (121) - INVALID DEV-NAME OR MOUNTED DEVICE

Using the SHLABEL command, the user has tried to display the descriptor of a volume already logically connected with the command MNT.

MOS-CMD (122) - PATHNAME TOO LONG

The path name specified in the command is longer than allowed.

MOS-CMD (123) - INVALID PATHNAME

The path name specified in the command is not correct.

MOS-CMD (125) - RECORD TOO LONG

The record to display is too long.

MOS-CMD (126) - FILE IN USE

The file to be displayed is opened for writing by another system user.

MOS-CMD (127) - INCOMPATIBLE FILE TYPES

The type of organisation of the file and its access mode are incompatible.

MOS-CMD (128) - INVALID FILE TYPE

The requested operation is incompatible with the type of the file system item specified in the command.

MOS-CMD (129) - xx...x FILE TYPE IS NOT CORRECT

The file "xx...x" is not a byte-stream file.

MOS-CMD (131) - INVALID NUMERIC KEY

A non-numeric value has been entered as key parameter of a command.

MOS-CMD (134) - BAD FILENAME

The file to pack or unpack is not a byte-stream file.

MOS-CMD (135) - BAD DESTINATION

The path name given for the directory that is to contain the file to pack or unpack does not identify a directory.

MOS-CMD (136) - PRIVILEGED COMMAND

The command entered can only be performed by the system administrator (ROOT).

MOS-CMD (137) - FILE ALREADY EXISTS

The file created by the login operation exists already.

MOS-CMD (138) - ILLEGAL DESTINATION

The directory specified as user home directory is invalid.

MOS-CMD (139) - ERROR IN GLOBALNAME

Error in the name of the global resource defined.

MOS-CMD (140) - NAME NOT FOUND

The name of the file, directory, or volume specified in the command does not exist.

MOS-CMD (164) - ERROR IN PARAMETER

A command parameter is missing or specified incorrectly.

MOS-CMD (165) - ERROR IN CONNECT

An error occurred when connecting a file system item.

MOS-CMD (166) - ERROR IN GET FILE TYPE

An error occurred when reading the file type.

MOS-CMD (167) - ERROR IN GET FILE INFO

An error occurred when reading the structure containing file information.

MOS-CMD (168) - ERROR IN CONNECT PARENT DIR

An error occurred during the connection to the father directory.

MOS-CMD (169) - ERROR IN CONNECT PDDTABLE

An error occurred during the connection to the Permanent Dataset Descriptor (PDD) table.

MOS-CMD (171) - SECURITY VIOLATION

The access rights of the file system item do not allow the requested operation to be carried out.

MOS-CMD (172) - NO REMOTE ACTION - INVALID OPERATION

The remote operation requested is invalid, as the system is not distributed.

MOS-CMD (1121) - CLEARDIR : CANNOT CLEAR . OR ..

The directories "." and ".." cannot be used as parameters of the CLEARDIR command.

MOS-CMD (1204) - GENERATING LOOP

Using the command CPTREE, an attempt has been made to copy a directory onto itself or one of its sub-directories.

MOS-CMD (1510) - ERROR READING xx...x

An error occurred while reading the file "xx...x".

MOS-CMD (1516) - ERROR SETTING WORKING DIRECTORY

An error occurred while connecting to the working directory.

MOS-CMD (1517) - ERROR CONNECTING xx...x

An error occurred while connecting to the file "xx...x".

MOS-CMD (1518) - ERROR OPENING xx...x

An error occurred while opening the file "xx...x".

MOS-CMD (1520) - CANNOT COMPARE INPUT FILES

The two input files cannot be compared because their structure is different from that of normal byte-stream files.

MOS-CMD (1521) - CANNOT COMPARE TWO VOLUMES

The DIFF command cannot compare two volumes.

MOS-CMD (1522) - CANNOT COMPARE TWO DIRECTORIES

The DIFF command cannot compare two directories.

MOS-CMD (1523) - FILES CANNOT BE ANALYSED: THEY ARE "INCOMPATIBLE"

The two files cannot be compared because they are file system objects of different types.

MOS-CMD (1525) - DIFF UTILITY : ABORT

An error occurred during the execution of DIFF which does not permit normal termination.

MOS-CMD (1805) - GETLOGUSER ERROR N. n

Error while reading the login data base.

MOS-CMD (1851) - WARNING: FAMILY n NOT KILLED

This family cannot be found, and therefore cannot be deactivated.

MOS-CMD (1853) - WARNING: CANNOT KILL FAMILY n

The family specified cannot be deactivated because it is the father of the family which has called the KILLFAM command.

MOS-CMD (1854) - WARNING: GRANDPA FAMILY CANNOT BE KILLED

The Grandpa family has been specified, and this cannot be deactivated.

MOS-CMD (1913) - ERROR IN CONNECT

An error occurred during the connection to the file.

MOS-CMD (1914) - ERROR IN OPENBYTE

An error occurred while opening the file.

MOS-CMD (1915) - ERROR IN READBYTE

An error occurred while reading the file to display.

MOS-CMD (1916) - ERROR IN CLOSEBYTE

An error occurred while closing the file displayed.

MOS-CMD (1917) - ERROR IN DISCONNECT

An error occurred while releasing the connection to the file.

MOS-CMD (2101) - SORRY ... UNABLE TO SORT

The contents of the directory cannot be displayed using the sort options of the LS command, as there is not enough space on disc for allocating the temporary file used by the system.

MOS-CMD (2105) - xx...x : INVALID PATHNAME

The path name specified in the command is invalid.

MOS-CMD (2201) - xx...x : INVALID PATHNAME

The path name specified in the command MKALIAS is invalid.

MOS-CMD (2202) - xx...x : NO REMOTE ALIASING ALLOWED

It is not possible to perform an alias operation on a remote file.

MOS-CMD (2861) - DEVICE IS NOT A HARD DISK

Dump space can only be allocated on a hard disc.

MOS-CMD (3103) - IN USE

The file to be displayed using the MORE command is already open for writing by another system user.

MOS-CMD (3200) - WARNING: MTA ** PRIVILEGED COMMAND **

MTA can only be executed by the system administrator.

MOS-CMD (3201) - MWS NOT AVAILABLE

The master work station does not exist.

MOS-CMD (3202) - NO REQUESTS TO MASTER WORK-STATION

No message has been sent to the master work station.

MOS-CMD (3203) - ERROR IN SENDING REPLY

Self-explanatory.

MOS-CMD (3204) - CANNOT ACCESS MWS QUEUE

The master work station is defined, but it is not possible to access the message queue.

MOS-CMD (3304) - xx...x : IS A PACKED FILE

The file "xx...x" is already packed.

MOS-CMD (3305) - xx...x : TRIVIAL FILE

The frequency of the characters contained in the file to be compacted is such that it does not allow compaction.

MOS-CMD (3306) - xx...x : NOT PACKED

The file "xx...x" cannot be packed for one of the following reasons:

- it is empty
- the result of packing would only recover less bytes than the block size.

MOS-CMD (3307) - xx...x : NOT BST

The file to pack is not a byte-stream file.

MOS-CMD (3308) - xx...x : HUFFMAN TREE HAS TOO MANY LEVELS

The file "xx...x" cannot be packed using the HUFFMAN method, as the HUFFMAN tree has too many levels.

MOS-CMD (3309) - xx...x.R: WRITE ERROR

An error occurred when rewriting the packed file.

MOS-CMD (3601) - REMOVE : CANNOT REMOVE . OR ..

The directories "." and ".." cannot be removed using the REMOVE command.

MOS-CMD (3651) - RENAME : CANNOT RENAME . OR ..

The RENAME operation cannot be performed on the directories "." and ".."

MOS-CMD (3671) - ERROR IN CONNECT STDOUT

An error occurred during connection to the standard output.

MOS-CMD (3672) - ERROR IN GETTYPE FOR STDOUT

An error occurred when searching for the standard output.

MOS-CMD (3681) - ERROR READING EXTENT TABLE

An error occurred while reading the extent table.

MOS-CMD (3682) - ERROR READING A PDD

An error occurred while reading the Permanent Dataset Descriptor (PDD) table.

MOS-CMD (3683) - ERROR OPENING A BST

An error occurred while opening a byte-stream file.

MOS-CMD (3684) - ERROR READING A VOL_DESC

An error occurred while reading a volume descriptor.

MOS-CMD (3725) - ERROR OPENING A BST

An error occurred while opening a byte-stream file.

MOS-CMD (4101) - PHYSICAL SCREEN SIZE CANNOT BE CHANGED

The screen size cannot be changed because of its physical characteristics: for instance on an M24 terminal or for a graphics screen.

MOS-CMD (4201) - xx...x : INVALID OPERATION

The file "xx...x" specified in the command SHALIAS is not an alias file.

MOS-CMD (5003) - xx...x : READ ERROR

An error occurred when reading the file to unpack.

MOS-CMD (5004) - xx...x : IS NOT A PACKED FILE

The file "xx...x" is not a packed file.

MOS-CMD (5005) - xx...x : UNPACKING ERROR

An error occurred while unpacking the file.

MOS-CMD (5006) - xx...x : WRITE ERROR

An error occurred when rewriting the file to unpack.

MOS-CMD (5007) - xx...x : NOT CREATED

The file "xx...x" cannot be created.

MOS-CMD (5051) - NO ENTRIES

There are no users in the "login" data base.

MOS-CMD (5052) - DATA BASE IN USE

The "login" data base is being used by another user.

MOS-CMD (5053) - ERROR OPENING FILE

Self-explanatory.

MOS-CMD (5054) - ERROR READING FILE

Self-explanatory.

MOS-CMD (5055) - ERROR WRITING TMP FILE

Error while writing the temporary file.

MOS-CMD (5056) - USER NOT FOUND

The user name given does not exist: check that it has been typed correctly.

MOS-CMD (5057) - GROUP NOT FOUND

The group name given does not exist: check that it has been entered correctly.

MOS-CMD (5058) - NAME ALREADY EXISTS

The user or group name to be added exists already.

MOS-CMD (5060) - TOO MANY USERS IN THE GROUP

There are too many users in the group, so no more can be added.

MOS-CMD (5061) - LOGIN AND USER NAME CANNOT BE MODIFIED

It is not possible to modify a user login, because that user is connected to the system.

MOS-CMD (5062) - GROUP NOT EMPTY

The specified group cannot be deleted because there are still users in it. All the users in the group must be deleted before it can be removed.

MOS-CMD (5073) - INVALID OPERATION ** PRIVILEGED COMMAND **

Only the system administrator can execute the requested operation.

MOS-CMD (5074) - LINE DOWN: OPERATION ABORTED

The line went down during an operation on the login data base.

MOS-CMD (5075) - YOU MUST REMOVE USER/GROUP FROM MACHINE WHERE CREATED

A group or user can only be removed from the machine where it was created.

MOS-CMD (5102) - xx...x : NOT FOUND

The file searched for has not been found.

MOS-CMD (5401) - WARNING : USER NOT LOGGED

The user recipient of the message is not connected to the system.

MOS-CMD (5402) - USER NOT FOUND

The user recipient of the message is not configured in the system.

MOS-CMD (5451) - ERROR READING A PDD

An error occurred while reading the Permanent Dataset Descriptor (PDD) table.

MOS-CMD (5452) - ERROR READING EXTENT TABLE

An error occurred while reading the extent table.

MOS-CMD (5453) - ERROR IN GETNAME

An error occurred when searching for the name of the file.

MOS-CMD (5454) - ERROR OPENING DIRECTORY

An error occurred while opening a directory.

MOS-CMD (5455) - ERROR READING DIRECTORY

An error occurred while reading a directory.

MOS-CMD (5458) - ERROR OPENING PDDTABLE

An error occurred while opening the Permanent Dataset Descriptor (PDD) table.

MOS-CMD (5459) - ERROR IN OPTION

The option specified in the WSTAT command is incorrect.

MOS-CMD (5463) - ERROR OPENING A BST

An error occurred when opening a byte-stream file.

MOS-CMD (5464) - ERROR READING A BST

An error occurred when reading a byte-stream file.

MOS-CMD (5601) - COPY : CANNOT COPY . OR ..

The COPY command cannot be performed because the current directory "." or the father directory ".." has been defined as the source directory.

MOS-CMD (6001) - NO REMOTE ACTION - INVALID OPERATION

The access rights of remote file systems objects cannot be modified.

MOS-CMD (6101) - ERROR IN GET USER NAME PARAMETER

An error occurred while searching for the user name.

MOS-CMD (6102) - ERROR IN GET USER FUNCTION

An error occurred while searching for the user function.

MOS-CMD (6103) - NO REMOTE ACTION - INVALID OPERATION

The owner of remote file system items cannot be changed.

B. GENERAL LIMITATIONS AND CHARACTERISTICS

This appendix describes the limits related to hardware and software objects referred to by Shell commands.

OBJECT	LIMITATIONS
allocation unit	value in the range 1 to $2^{31}-1$ rounded up to the nearest multiple of 512.
MCL command line	maximum of 160 characters (including CR), on both screen and system line.
object name (file/directory/volume)	maximum of 14 characters (12 for positional or keyed files).
path name	maximum of 60 characters. If remote, the machine identifier must also be included in this maximum.
peripherals	
FD (8" floppy disc)	1 Mbyte capacity (984,576 bytes): - double side and double density - 75 cylinders per side (74 for the user) - 2 tracks per cylinder - 26 sectors per track - 256 bytes per sector

(cont.)

OBJECT	LIMITATIONS
MF (5" floppy disc)	320 Kbytes capacity (273,408 bytes): - double side and double density - 33 cylinders per side (32 for the user) - 2 tracks per cylinder - 16 sectors per tracks - 256 bytes per sector 1 Mbyte capacity (984,576 bytes): - double side and double density - 75 cylinders per side (74 for the user) - 2 tracks per cylinder - 26 sectors per track - 256 bytes per sector
HD (Hard Disc)	These must be used on the drives specified below, with the disc controller (interface) indicated.
SASI (HDA 3319) 14 Mbytes (14,689,280 bytes for the user) * HDU 3455 (integrated) HDU 3455 (external)	
HDC 3555 18 Mbytes (17,397,760 bytes for the user) * HDE 3511	

OBJECT	LIMITATIONS

ST506 (HDC 3544)
10 Mbytes (9,721,156 bytes for the user) *
HDU 3407 (integrated, SLIM)

20 Mbytes (19,919,360 bytes for the user) *
27 Mbytes (27,965,440 bytes for the user) *
HDU 3425 (integrated)

40 Mbytes (41,234,944 bytes for the user) *
HDU 3449 (integrated)

65 Mbytes (65,994,752 bytes for the user) *
HDU 3465 (integrated, WREN)
HDU 3666/67 (integrated, WREN)
HDU 3668/69 (CAB 3558, WREN)

SMD (HDC 3527)
60 Mbytes (61,523,456 bytes for the user) *
HDU 3560/78 (CAB 3558)
HDU 3565/66 (MTU 3541)

120 Mbytes (123,238,400 bytes for the user) *
275 Mbytes (274,676,160 bytes for the user) *
HDU 3516/79 (CAB 3558)

HDC 7050 (ESDI disc-controller)
140 Mbytes (140,000,000 bytes for the user) *
HDU 5451

* From this figure subtract 16,384 bytes if the HD contains the environment activator (LDHMU2) and/or an extra 1.5 Mbytes if the diagnostic programs are also present.

(cont.)

OBJECT	LIMITATIONS

MTU (Magnetic tape)

(MTU 3541)

```
width    =  1/2"
density  =  1600 bpi (bits per inch)
capacity =  40 Mbytes (start/stop mode)
            20 Mbytes (streaming mode)
```

SCT (Streaming Cartridge Tape)

(STS 3419)
(STS 3554)

```
width    =  1/4"
capacity =  19 Mbytes
tracks   =  4
blocks   =  9240 (per track)
bytes    =  512 (per block)
recording mode  = single/double
```

Note: STS 3554 is integrated on M30 and M34 machines.

SCT referred to as SCT5:

(STS 5437, for SLIM drives, which can also handle non SCT5 cassettes)

```
width    =  1/4"
length   =  450/600 feet
density  =  8000 bpi (bits per inches)
capacity =  45/60 Mbytes
tracks   =  9
bytes    =  512 (per block)
```

OBJECT	LIMITATIONS
remote commands	Commands which operate on remote resources are: --- Shell commands--- BATCH FILETAR MKKEYED SHALIAS BMQ LIST MKPOS SHDIR CALLMWS LPQ MKVOL SPOOL CHTYPE LPR MTA WHEREIS CLEARDIR LS MORE COPY MKALIAS PRY CPTREE MKBST REMOVE DIFF MKDIR RENAME EXIST MKGLOB SETINFO --- MCL built-in commands --- BM DISCON SETWDIR CONN EDPARSE SHWDIR CONSP READ SHCON --- system maintenance commands --- CSIZE LISTMSG
users	The USR group, which contains all the user names, must be created in the directory /IPL. The home directory for each user is "/IPL/USR/user-name".
workstation identity	Local workstations are given a single character identifier at configuration time. A maximum of 32 workstations is possible, in the range A to Z and 0 to 5, with the prefix TTY. Terminals may be configured as "terminals with dynamic login". These are not "hard-wired" to the system, and are not "known" to the system until they are logged in, when they are given an identifier in the range 01 to 32. This identifier varies depending on the number of terminals already logged in. For example, if six such terminals are already logged in, the next to login is given the identifier "07". The prefix is RT and applies to PCs connected to a local network.

C. MAGNETIC TAPE CHARACTERISTICS

INTRODUCTION

Tapes are amongst the most commonly used storage media. This appendix is intended to help users understand some of their working modes.

TAPE TYPES

Two types of tape are used on Olivetti Systems:

- SCT (Streaming Cartridge Tape)
- MTU (Magnetic Tape Unit).

Their modes of operation differ, as the descriptions below show, but their contents have much in common.

CONTENTS COMMON TO BOTH TYPES

TAPE MARKS AND LABELS

All labels should conform to ECMA 13 standard.

The most essential "punctuation mark" on a tape is the delimiter called a tape mark. Whenever the system reads a tape and meets a tape mark, it "knows" that it is either the beginning or the end of a dataset.

A dataset, by definition, may be a label created by the system, or a collection of data of use to the user.

Labels are in fact written to tape in groups enclosed by tape marks, as all datasets (see below).

Sometimes, complete label groups are only required at the beginning and end of tape. In such cases, the tape is declared when created as being intended for "unlabelled" mode (see below), but tape marks must still appear where labels would have appeared normally.

Such label groups are written at the beginning of the tape, at the beginning and at the end of each dataset, and at the end of the tape. Those at the end of the tape also show whether the volume is continued on another tape (see below).

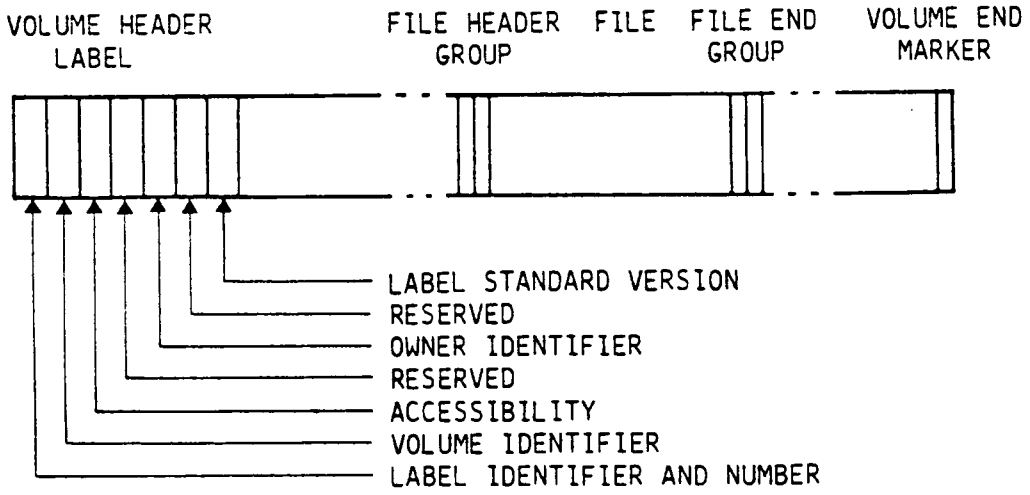


Fig. C-1 Logical Organisation of a Magnetic Tape

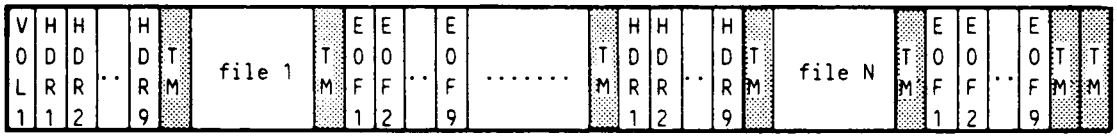


Fig. C-2 Organisation of a Labelled Tape

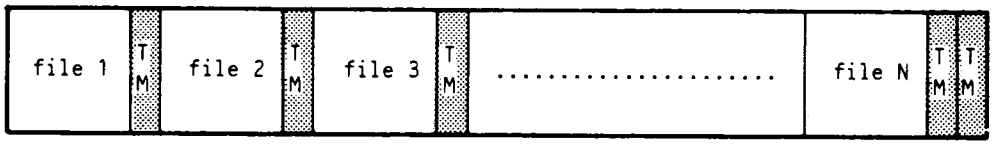


Fig. C-3 Organisation of an Unlabelled Tape

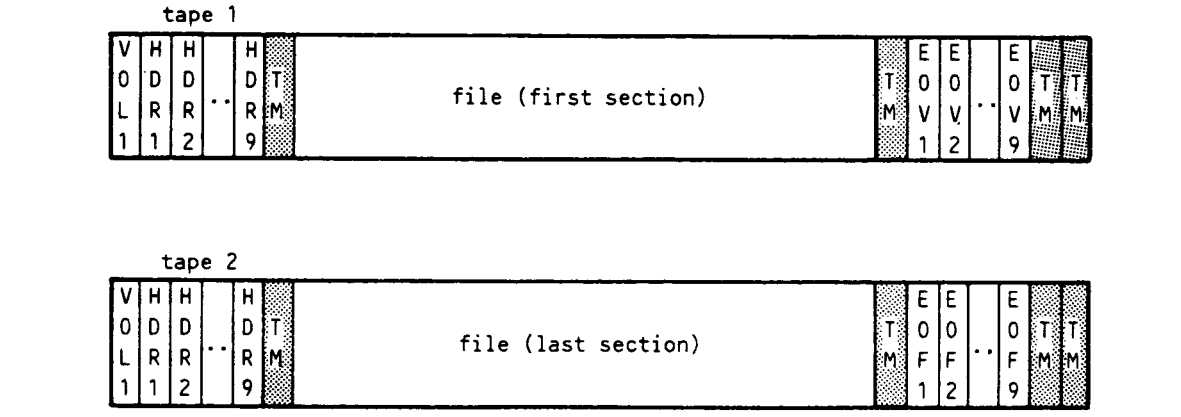


Fig. C-4 Labelling of a Multi-Tape File

The groups referred to most often are called (perhaps inaccurately, but conveniently) the "Tape Header Label" and the "Volume Header Label".

TAPE HEADER LABEL

The tape header label is the first label on the tape. It is created when the tape is initialised, and contains the following information:

- tape identifier
- tape owner
- multi-tape volume indicator
- double-recording indicator (SCT only)
- date of initialisation
- information about the data on the tape.

This information identifies the owner and describes the use of the tape.

VOLUME HEADER LABEL

A volume on tape is preceded by a label which contains:

- the volume name
- the volume size
- other information.

The tape header label, volume descriptors and volume itself are separated by tape marks (TM).

SCT (STREAMING CARTRIDGE TAPE)

The SCT consists of 450 feet (136.26 metres) of 1/4 inch high density tape, contained in a cassette or cartridge, holding 19 Mbytes of data. Each of its 4 tracks (0 to 3) must be able to hold at least 9240 blocks of 512 bytes each. If any track cannot satisfy this requirement, the tape must not be used.

Each byte of data is written to an MTU ACROSS the width of the tape, a bit being written to each of its tracks simultaneously. On the SCT, data is written ALONG a track in the same way as data is written to a disc track, that is, bytes of records are written one bit following the other, not side by side.

At the physical end of the tape, writing continues onto track 1 back at the beginning of the tape, then back again on track 2, and so on. (See figure below.)

Write protection

The SCT has a write-protect (that is write-prevent) knob on its case. Turning this until its arrow points to "SAFE" prevents writing to the tape.

Single/double recording

The SCT is capable of single or double-recording mode.

Single-recording can store a maximum of 18.8 Mbytes and double-recording 9.4 Mbytes of data, (see Fig. C-5 below).

Double-recording means that back-up copies of the volumes can be stored on the same tape.

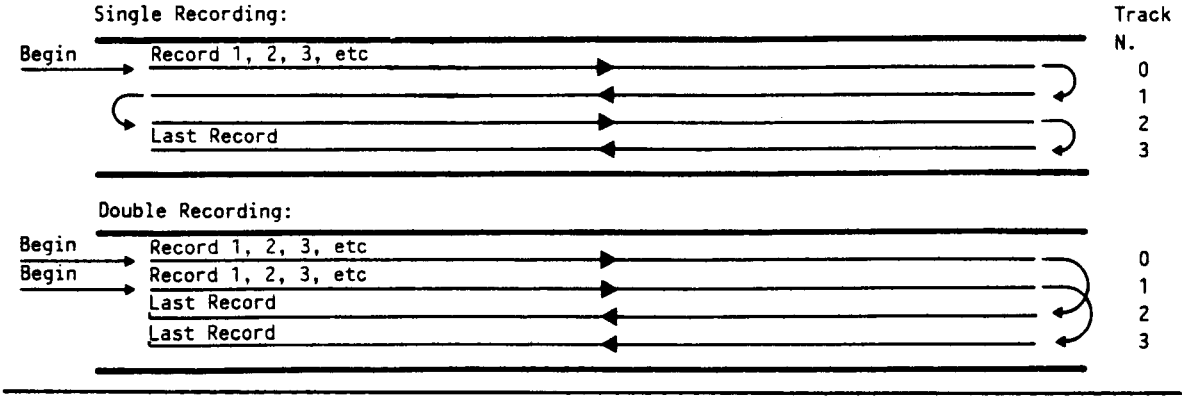


Fig. C-5 Single and Double-Recording

Do not double-record on multi-volume tapes. Back-up copies for these should be made on separate tapes.

Note

1. If the SCT is winding and the drive lights are flashing, do not attempt to remove it from the drive.

2. More than one volume may be saved on a single SCT.

3. The save of a large volume may be continued on other SCTs.

4. If required, a volume may be recorded twice on the SCT.

5. Commands which have a verify function for an SCT can produce a display of labels as shown below.

```
                    TAPE HEADER LABEL
TAPE IDENTIFIER: zz...z
OWNER IDENTIFIER: yy...y
TAPE MODE: S/M
RECORDING MODE: S/D
CREATION DATE: day month time year

                    TAPE  VOLUMES  LIST

NAME: volume name
SIZE: nn...n SECTORS, bb...b BYTES
SECTION: sequence number
SAVED ON: day month time year
TYPE: LOGICAL/FIELD NOT INITIALIZED
       .              .              .
       .              .              .
       .              .              .
                    END OF VERIFY
```

The display shows: the tape identifier (zz...z), the tape owner (yy...y), the multi-tape indicator (S/M), the double or single recording indicator (S/D), the creation date and the list of volumes on the tape.

For each volume the display shows: the name (volume name), the length in bytes (bb...b) and sectors (nn...n), the sequence number of the tape (secnum), the creation date and the type of volume (logical, or created with a previous release of VOLSR).

MTU (MAGNETIC TAPE UNIT)

The physical characteristics of an MTU differ from those of an SCT. 1200 feet of 1/2" wide tape are stored on a single reel. It has 7 or 9 tracks, and each byte of data is stored across the width of the tape, one bit per track, at a density of 1600 bpi (bits per inch).

Write Protection

The tape data is protected against overwriting/deletion by removal of a plastic ring from a groove around its centre (or hub) in one of its sides (flanges). To allow data to be written on the tape, a ring must be placed in the groove.

Organisation

The logical organisation of the MTU is the same as that of the SCT (see above "Contents Common to Both Types").

Tape marks and labels (e.g. tape and volume headers) are provided for exactly the same reasons and at the same positions relative to the datasets.

Datasets can be written as often as necessary to an MTU, and they may be continued on another. Back-ups of an MTU should always be made onto another MTU.

TAPE OPERATING SPEEDS

M30 and M40 machines can handle a tape only at 25 inches per second. Other hardware can handle tapes faster, if other conditions are also satisfied (see table below).

MODE	SPEED (inches/second)	USAGE (See 'Optimisation' also).
START/STOP	LOW	By default. Only for smallest file system objects. Available on all machines.
STREAMING	NORMAL (25 ips.)	Selected by user or default. Available on all machines. Where file system object < 100K.
STREAMING	HIGH (100 ips.)	Selected by user. For simple (not complex) file system objects, of files of 100K bytes or more. Only available on M44/M54/M60/M64, with hard disc and SMD or WREN controller.

Tab. C-6 Tape Speed Selection Criteria

Streaming mode handles tapes faster, where the machine and conditions allow.

In general, if streaming mode is suitable for a certain process, the system will enter that mode automatically.

Optimisation:

These notes apply to the operation of FILETAR, but they also serve to illustrate streaming mode in practice.

- The save operations specifying several file system objects at once produce more efficient tape writing. However, it is important to ensure that the "fastest" objects are dumped first. Indeed, as soon as it meets a "slow" object, FILETAR leaves streaming mode, continuing in START/STOP mode for all the subsequent dumps, even if some of them could be handled at a higher speed.

- Restore operations can also benefit from streaming mode speeds, because file objects need not be restored in the order in which they appear on tape. So the 'fastest' object can be handled first, even if it is the most distant on the tape.

- The UNLABEL option significantly speeds up tape operations, especially in handling highly structured file objects.

- For recursive files, the entire contents of a specified subdirectory, down to its lowest level, are saved to tape before starting on the next directory.

D. FLOPPY DISC CHARACTERISTICS

INTRODUCTION

Discs are among the most commonly used storage media. This appendix is intended to help users understand some of their working modes.

FLOPPY DISC DESCRIPTIONS

Olivetti systems use floppy discs of two sizes (physical diameters):

- 8 inches (8") discs
- 5 inches discs, (5", or more accurately, 5 1/4").

8 INCHES DISCS

These may be single or double-sided, and single or double density. Other data is shown in the table below.

TYPE	DISC CAPACITY	CYLINDERS PER DISC	TRACKS PER CYLINDER	SECTORS PER TRACK	BYTES PER SECTOR	BYTES PER TRACK
8"	1 Mbyte (984,576 bytes)	74	2	26	256	6656

5 INCHES DISCS

Only the double-sided variety may be used, but these may be single or double density. Other data is shown below.

TYPE	DISC CAPACITY	CYLINDERS PER DISC	TRACKS PER CYLINDER	SECTORS PER TRACK	BYTES PER SECTOR	BYTES PER TRACK
5"	320 Kbytes (273,408 bytes)	33	2	16	256	4096
	1 Mbyte (984,576 bytes)	74	2	26	256	6656

DISC ORGANISATION AND LABELS

INITIALISATION

Initialising a disc is the first step in making a "virgin" disc. The initialisation process writes the volume header label on the disc, containing identification details provided by the user. On Olivetti discs, the volume header label is written on cylinder 0. Any existing volume header is overwritten, and any data stored anywhere on the disc is logically deleted.

CYLINDER 0 (INDEX CYLINDER)

On a double-sided disc, cylinder 0 consists of two tracks: one track 0 on each side.

Cylinder 0 is the first area of a disc to be written upon. As the name suggests, it is reserved for system use, to record descriptions and locations of the contents of the disc in an index.

Whenever an object is added to the disc, its description is written in the index: the descriptive details come from the user, and the details of its location (address) are added by the system.

To retrieve data stored on the disc, the file handling system looks in the index to ascertain its presence and location on the disc, thus enabling direct access to the data, which is stored on the other cylinders.

SIDE	SECTORS	USE
0	01-04	Reserved for system use.
0	05	Error map label.
0	06	Reserved for future standardisation.
0	07	Volume label.

LABEL	CHARACTER POSITIONS	DESCRIPTION	CONTENTS
VOLUME	05 10	Volume identifier	maximum of 6 characters
"	38-51	Owner identifier	maximum of 14 characters
"	72	Code indicator	ASCII = "A", EBCDIC = "E"
"	80	Label type	ECMA = "3", IBM = "W".
FILE	44	Interchange type indicator	ECMA = " ", IBM = "H"
"	48-53	Creation date	maximum of 6 characters
"	67-72	Expiry date	maximum of 6 characters

Fig. D-1 Location of the Information on Cylinder 0

TAPE MARK

One of these is required to begin and end every dataset. It is used for the same purpose on disc as on tape. The first tape mark is written on disc or tape at initialisation time, followed immediately by the volume header. Thereafter, a tape mark is written by the system whenever appropriate.

Note that a "dataset" may be one of the various kinds of labels created by the system, as well as a set of data provided entirely by the user.

OTHER CYLINDERS

These are used for storing the data. On a double-sided disc there are two tracks for each cylinder.

RECURSIVE ORGANISATION

Data is usually stored "recursively" on disc. This term "recursive" may be used to describe a data structure like a directory, containing files, and also sub-directories which each contain more files and "sub-sub-directories", and so on.

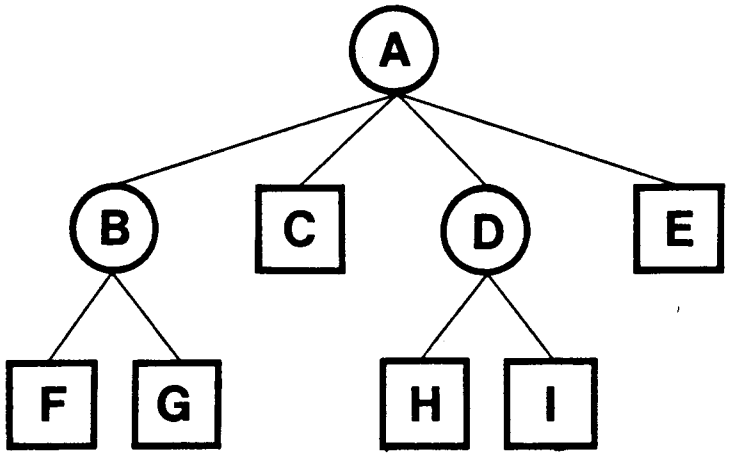


Fig. D-2 Recursive Organisation of Disc Data

MULTI-FLOPPY ORGANISATION

In a multi-floppy operation the object occurrence number is global to all the floppy discs used.

Let us suppose that there are 8 occurrences (versions) of file A on 3 floppy discs, as shown below, and the one shown with "^" has to be restored.

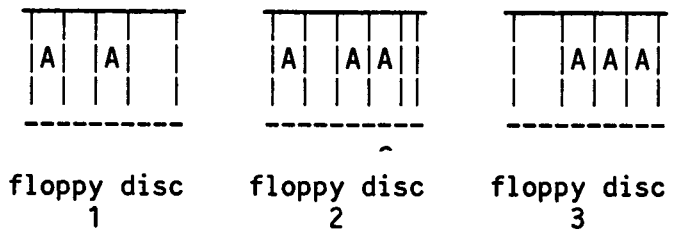


Fig. D-3 Retrieval from Multiple Occurrences

If floppy disc 1 is inserted before floppy disc 2, the user must identify the required file as occurrence number 5. If floppy disc 2 is inserted first, then the occurrence number is 3.

A file system object may begin on one floppy and end on another, as above.

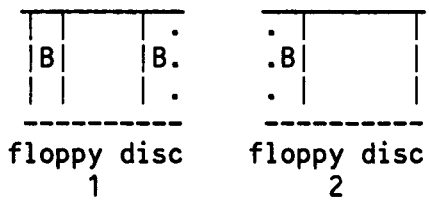


Fig. D-4 Disc file Continuation

Here, file B occurs twice on floppy disc 1, the second occurrence ending on floppy disc 2. But, although parts of B may be found in 3 different places, the system recognises only the two occurrences of the file, which are those on floppy disc 1.

DISPLAY OF CONTENTS

The contents of a floppy disc may be displayed as shown below.

```
                -- GLOBAL LABEL LIST --

SEQUENCE NUMBER: nn...n
NAME: xx...x
OWNER : identifier

                -- FLOPPY CONTENTS --

NAME: file name
TYPE: VOLUME/DIRECTORY/BYTESTREAM/POSITIONAL/KEYED
SIZE/ENTRY CONTENT(S): nn...n
SECTION NUMBER: secnum
FLOPPY LENGTH: mm...m
SAVED ON: time date
```

Fig. D-5 Illustration of a Floppy Disc Label

From the above display, the user can check the volume header of a floppy disc to find the sequence number, the disc identifier, the name of the owner, the name, type and creation date of all file system objects on the disc, as well as the object size or, for a directory, the number of entries contained. "mm...m" defines the length of the object or of the part of it present on the floppy disc.

DISC STANDARDS AND COMPATIBILITY

8" floppies and 5" floppies are labelled according to the standards ECMA 58 and ECMA 67 respectively. This latter is the equivalent of Olivetti Standard 17.

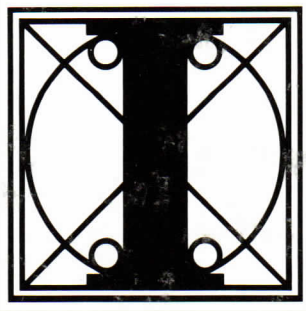
Olivetti utilities are designed to write labels and data to disc in such a way that, if required, they can be transferred to other systems which observe the same standards. Only data files can be converted, not program files or packed data files.

In particular, data exchange with IBM machines has been catered for (on 8" floppies). Conversion from Olivetti to IBM format (i.e. ASCII to EBCDIC) is easy: it is only necessary to specify the required data format when producing the exchange disc.

Even the M20 can read and write files written to ECMA 67 standard. The M20 normally uses a non-ECMA format, but it has a conversion program.

Differences still exist between Olivetti and IBM discs: whilst Olivetti machines fill unused floppy sectors with binary 0, IBM fill the first 80 characters of empty sectors with blanks (hexadecimal EBCDIC 40), and the remainder with binary 0. Furthermore, 8" discs produced to the Olivetti standard have 74 cylinders available for data, whereas the IBM format makes 73 cylinders available to the user.

Printed in Italy



olivetti