

**QUICK REFERENCE  
GUIDE**

**HP 1000**



**HEWLETT  
PACKARD**



# **RTE-6/VM QUICK REFERENCE GUIDE**



**HEWLETT  
PACKARD**

**DATA SYSTEMS DIVISION  
11000 WOLFE ROAD  
CUPERTINO, CALIFORNIA 95014**

**MANUAL PART NO. 92084-90003  
Printed in U.S.A. December 1983  
E1283**

# PRINTING HISTORY

New editions are complete revisions of the manual. Update packages contain replacement pages or write-in instructions to be merged into the manual by the customer. Manuals will be reprinted as necessary to incorporate all prior updates. A reprinted manual is identical in content (but not in appearance) to the previous edition with all updated incorporated. No information is incorporated into a reprinting unless it appears as a prior update. The edition does not change.

Second Edition ..... Dec 1983    CI file system added.

## NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another program language without the prior written consent of Hewlett-Packard Company.

Copyright © 1983 by HEWLETT-PACKARD COMPANY

# TABLE OF CONTENTS

	SECTION
MANUAL CONVENTIONS AND BOOT UP PROCEDURE .....	A
CI COMMANDS .....	B
FMGR COMMANDS .....	C
SYSTEM AND BREAKMODE COMMANDS .....	D
EDIT/1000 COMMANDS .....	E
LINK .....	F
UTILITIES .....	G
EXEC CALLS .....	H
CI FILE HANDLING .....	I
FMGR FILE HANDLING .....	J
GASP, ACCOUNT, BATCH AND SPOOLING COMMANDS .....	K
SMP CALLS AND VMA/EMA ROUTINES .....	L
TABLES .....	M
ERROR CODES .....	N



## CONVENTIONS USED IN THIS MANUAL

CONTENT	PAGE
COMMAND SYNTAX .....	A-2
FILE DESCRIPTORS .....	A-5
BOOT UP PROCEDURE .....	A-7

## CONVENTIONS USED IN THIS MANUAL

### COMMAND SYNTAX

This manual uses the following conventions to describe command syntax:

<b>CAPITAL LETTERS</b>	Commands or parameters that must be entered exactly as shown are in capital letters; however, CI always accepts lowercase input.
[ ]	Optional parameters are in brackets. If additional parameters follow an omitted parameter, commas must be used as placeholders.
< >	Parameters enclosed in angle brackets are optional in some cases and required in others.
	Parameter choices are separated by vertical bars.
,	Delimiters between commands and parameters are commas or spaces.
<b>lowercase letters</b>	Terms that must be replaced by actual parameters (variables) are in lowercase letters.
<b>single underline</b>	Single underlined parameters have values returned by the system.
<b>double underline</b>	Double underlined parameters have values returned by the system in some cases, and are user supplied in other cases.

The following definitions apply throughout this manual. Terms not described below are defined under each command description.

<b>prog</b>	One- to five-character program name. Can be followed by slash (/) and session ID (provided by the command WH,SE). Examples: A, PROGA, TIMER, LRUN/3. (In the last example, 3 is the session ID.)
<b>lu</b>	Logical unit number, in the range 0 to 255, inclusive. Refers to a physical I/O device. LU 1 always refers to the user terminal. LU 0 refers to "the bit bucket," a nonexistent device (unwanted data can be sent to LU 0).
<b>file</b>	File descriptor, which unambiguously specifies a single file.

namr = filedes = name [:security [:cartridge [:type [:file size  
[:record size]]]]]

or

namr = logical unit number

where:

<b>security</b>	<ul style="list-style-type: none"> <li>&lt;0 Write and read protected</li> <li>0 Not protected (default)</li> <li>&gt;0 Write protected</li> </ul>
<b>cartridge</b>	<ul style="list-style-type: none"> <li>&lt;0 lu number</li> <li>0 First available cartridge (default)</li> <li>&gt;0 FMGR cartridge reference number</li> </ul>
<b>file type</b>	<ul style="list-style-type: none"> <li>0 Non-disc file</li> <li>1 128-word record length, random access</li> <li>2 User selected record length, random access</li> <li>3 (and greater) variable record length, sequential access</li> <li>4 Source program</li> <li>5 Relocatable program</li> <li>6 RTE load module</li> <li>7 Absolute program</li> <li>&gt;7 User defined</li> </ul>
<b>file size</b>	<p>Specified in blocks (2 sectors = 1 block = 128 words).</p> <ul style="list-style-type: none"> <li>+n = allocate n blocks.</li> <li>-n = allocate n 128 block multiples.</li> <li>-1 = allocate remaining space on cartridge.</li> </ul>
<b>record size</b>	Used only when file is type 2.



## CONVENTIONS USED IN THIS MANUAL

**mask** Mask field, a file descriptor that can include the two "wild card" characters "-" and "@" and a mask qualifier. Mask qualifiers are:

- a** access data range
- b** back-up files
- c** creation date range
- d** match any directory
- e** search every disc volume
- n** not (e)
- o** open files
- p** purged files
- s** search all subdirectories
- u** update date range
- x** files with extents

For mask qualifiers a, c and u, the date range is specified:

`.a[yymmdd.hhmmss]-[yymmdd.hhmmss]`

where yymmdd.hhmmss represents year, month, day, hours, minutes, seconds.

**file|lu** Either a file descriptor or a logical unit can be specified.

**mask|lu** Either a mask or a logical unit can be specified.

**param** One parameter is allowed.

**param\*n** As many as n parameters are allowed. In most applications, unspecified parameters default to zero or zero-length strings.

**prog|file** Either a program name or a file descriptor can be specified. Refer to the RU command for details.

## FILE DESCRIPTORS

A file descriptor has up to 63 characters and one of the following formats, depending upon the application (the first two are equivalent):

### CI FILE FORMAT:

/directory/subdirectory/filename:::type:size:recordlength

### COMBINED FORMAT:

subdirectory/filename::directory:type:size:recordlength

### FMGR FORMAT:

filename:securitycode:cartridge:type:size:recordlength



A filename has up to 16 characters, plus a type extension, and cannot include the characters "at sign" (@), minus (-), slash (/), period (.), or left bracket ([).

A type extension is separated from the filename by a dot (.) and has up to four characters. The standard type extensions are:

.cmd	command file
.dbg	debug file
.dat	data file
.dir	directory or subdirectory entry
.ftn	FORTRAN source file
.lib	indexed library of relocatables
.lod	LINK loader command file
.lst	listing
.mac	Macro source file
.map	loader map listing
.pas	Pascal source file
.rel	relocatable (binary) file
.run	program file
.snp	system snapshot file
.txt	text file
.sys	system file

## CONVENTIONS USED IN THIS MANUAL

File type is specified by an integer in the range zero to 255. Default is type three. Standard file types are:

- |         |  |   |
|---------|--|---|
| 0       | I/O device. No directory entry. Usually used as a program parameter. |  |
| 1       | Random-access file; 128-word records.                                |   |
| 2       | Random-access file; user-specified record length.                    |   |
| 3 and 4 | Sequential-access text file; variable-length records.                |  |
| 5       | Relocatable object code file.  |   |
| 6       | Executable program file.   |   |
| 7       | Absolute binary file.  |   |

File size is specified by an integer in the range  $-32768$  to  $32767$ , inclusive. A positive number allocates space in blocks (128 words each); a negative number allocates space in 128-block sections.

Record length (in words) must be specified for a type 2 file. For other file types, this field is ignored.

## BOOT-UP PROCEDURE

1. Select the S-register for display on the computer front panel.
2. Press CLEAR DISPLAY.
3. Set the S-register bits as follows:

Bits:	Enter:
0-2	Surface number of the disc where the RTE-6/VM system subchannel starts.
3-4	0 (reserved).
5	0 for standard boot-up.
6-11	Octal select code of the disc.
12	1 to indicate a manual boot from the S-register.
13	0 (reserved).
14-15	Loader ROM selection (number of the ROM cell containing the Disc Boot Loader).

4. Press STORE.
5. Press PRESET, IBL and PRESET (again) to load contents of Disc Loader ROM.
6. Press RUN.



# COMMAND INTERPRETER (CI) COMMANDS

CONTENT	PAGE
AS (ASSIGN PARTITION)* .....	B-3
BR (BREAK PROGRAM EXECUTION) .....	B-3
CL (LIST MOUNTED DISCS) .....	B-3
CN (CONTROL DEVICE) .....	B-4
CO (COPY FILES) .....	B-4
CR (CREATE FILE) .....	B-5
CRDIR (CREATE DIRECTORY/SUBDIRECTORY) .....	B-5
DC (DISMOUNT DISC VOLUME) .....	B-5
DL (DIRECTORY LIST) .....	B-6
EX (EXIT) .....	B-7
GO (RESUME SUSPENDED PROGRAM)* .....	B-7
IN (INITIALIZE DISC VOLUME) .....	B-7
LI (LIST FILES) .....	B-7
MC (MOUNT DISC VOLUME) .....	B-8
MO (MOVE FILES) .....	B-8
OF (STOP OR REMOVE PROGRAM)* .....	B-8
OWNER (DISPLAY OR MODIFY OWNER) .....	B-8
PR (CHANGE PROGRAM PRIORITY)* .....	B-9
PROT (DISPLAY OR MODIFY PROTECTION) .....	B-9
PU (PURGE FILES) .....	B-9
RN (RENAME FILE OR DIRECTORY) .....	B-9
RP (RESTORE PROGRAM FILE) .....	B-10
RU (RUN PROGRAM)* .....	B-10
SL (SPOOL COMMAND) .....	B-10
SS (SUSPEND PROGRAM)* .....	B-11
SZ (DISPLAY OR MODIFY PROGRAM SIZE)* .....	B-11
TM (DISPLAY OR SET SYSTEM CLOCK) .....	B-12
TO (DISPLAY OR MODIFY DEVICE TIME OUT) .....	B-12
TR (TRANSFER TO COMMAND FILE) .....	B-12

## COMMAND INTERPRETER (CI) COMMANDS

UN (UNLOCK SHAREABLE EMA PARTITION)* .....	B-13	
UNPU (UNPURGE FILES) .....	B-13	
UP (UP A DEVICE)* .....	B-13	
VS (DISPLAY OR MODIFY VIRTUAL EMA SIZE)* .....	B-13	☾
WD (DISPLAY OR MODIFY WORKING DIRECTORY) .....	B-13	
WH (SYSTEM STATUS REPORTING) .....	B-14	
WS (DISPLAY OR MODIFY VMA WORKING SET SIZE)* .....	B-14	
XQ (RUN PROGRAM WITHOUT WAIT) .....	B-15	☾
? (HELP) .....	B-15	☾
/ (COMMAND STACK) .....	B-15	
\$1-\$9 (VARIABLE PARAMETERS) .....	B-15	

\*These commands can be entered in response to a SYSTEM prompt.

**AS (Assign Partition)\***

**Purpose:** Assign a reserved partition to a program.

**Syntax:** AS prog partNum

**prog** Program name, up to five characters, session identifier optional.

**partNum** Partition number. PartNum = 0 removes the current assignment.

\*This command can be entered in response to a SYSTEM prompt.

**BR (Break Program Execution)**

**Purpose:** Set a flag to allow limited communication with a program.

**Syntax:** BR[ prog]

**prog** Program name. Default is last scheduled program.

**CL (List Mounted Discs)**

**Purpose:** Display all mounted disc volumes.

**Syntax:** CL



## COMMAND INTERPRETER (CI) COMMANDS

### CN[,namr[,function[,subfnctn]]]

Issue control request to non-disc device.

<b>namr</b>	Type 0 file name or lu (default=LU8).
<b>function</b>	Control code, mnemonic (for octal see EXEC 3 call). mnemonic
	RW    rewind (default=MT,CTU)
	EO    end-of-file
	TO    top-of-form (default=LP,CRT)
	FF    forward space file
	BF    backspace file
	FR    forward space record
	BR    backspace record
	LE    leader (default=paper tape punch)
<b>subfnctn</b>	Carriage control.
	+n    to space n lines before next print operation.
	-n    page eject on line printer or space -n lines on terminal.

### CO (Copy files)

**Purpose:** Copy one or more files between directories and/or I/O devices.

**Syntax:** CO mask|lu1 mask|lu2[ param]

**mask|lu1** Source file or device.

**mask|lu2** Destination file or output device.

**param** One of the following characters (default is A):

- A** ASCII records; no checksum.
- B** Binary absolute; checksum performed.
- D** Overwrite duplicate files.
- P** Purge source after copying.

## CR (Create File)

**Purpose:** Create a disc file.

**Syntax:** CR file

**file** File descriptor (up to 63 characters) in one of the following formats:

**STANDARD:**

/directory/subdirectory/fileName:::type:size:recLength

**COMBINED:**

subdirectory/fileName:::directory:type:size:recLength

**FMGR:**

fileName:sc:crn:type:size:recLength

Refer to Section A for more detail on file descriptors.

## CRDIR (Create Directory/Subdirectory)

**Purpose:** Create a global directory or a subdirectory.

**Syntax:** CRDIR directory[ lu]

**directory** Directory name (up to 63 characters).

The name can include an optional size subparameter specified in number of blocks with the format:

directory:::size

Default size is the track size of the disc, typically 48 or 64 blocks for a hard disc and 30 or 16 for a flexible disc. Directory size is extended as needed.

**lu** LU of volume on which to create global directory. Default is LU of working directory. Volume of the working directory is used. Ignored for a subdirectory.

## DC (Dismount Disc Volume)

**Purpose:** Dismount a disc volume.

**Syntax:** DC lu

**lu** LU number of the disc volume to be dismounted. Can be positive or negative.

## COMMAND INTERPRETER (CI) COMMANDS

### DL (Directory List)

**Purpose:** List files in a directory.

**Syntax:** DL[ mask[ options[ file|lu[ msc]]]]

**mask** Specifies files to be displayed. Default is all files in the working directory.

**options** Information to be shown for displayed files (can be listed without delimiters in any order).

**A** ACCESS time.

**B** Indicate files that have not been BACKED UP with an \*.

**C** CREATION time.

**F** FILE type.

**L** File LOCATION (block address on disc).

**M** MAIN file size in blocks, excluding extents.

**N** NUMBER of records.

**O** Display OPEN files.

**P** File PROTECTION in the form owner/other.

**R** Length (in words) of longest RECORD.

**S** File SIZE in blocks, including extents.

**T** Indicate TEMPORARY files.

**U** UPDATE time.

**W** Number of WORDS in file, up to EOF.

**X** Indicate files with EXTENTS.

**Y** Security code (FMGR files only).

**\*** Options F, W, N, S, X, and P.

**!** All options.

**+** Ascending sort by item specified.

**-** Descending sort by item specified.

**file|lu** File or LU where the DL output is to be stored.

**msc** Master security code for the system. Required when Y or ! options are specified.

**EX (Exit)**

**Purpose:** Terminate CI.

**Syntax:** EX

**GO (Resume Suspended Program)\***

**Purpose:** Resume execution of a suspended program.

**Syntax:** GO[ prog[ param\*5]]

**prog** Name of the suspended program.

**param\*5** Parameters to be passed to the program (only if the program has suspended itself).

\*This command can be entered in response to a SYSTEM prompt.

**IN (Initialize Disc Volume)**

**Purpose:** Prepare a blank disc volume for system use.

**Syntax:** IN lu[ blocks[ OK]]

**lu** LU number of the disc volume to be initialized.

**blocks** Number of blocks to be reserved at the beginning of the disc LU for the boot extension and diagnostics (default is zero).

**OK** Suppresses user prompt.

**LI (List Files)**

**Purpose:** List files to a device.

**Syntax:** LI file[ format[ line1[ line2]]]

**file** File to be displayed.

**format** One of the following:  
A ASCII (default for type 3 and 4 files)  
B Octal (default for all other file types)

**line1** First line (default = 1).

**line2** Last line (default = line1).

If both line1 and line2 are omitted, all lines are listed.

## COMMAND INTERPRETER (CI) COMMANDS

### MC (Mount Disc Volume)

**Purpose:** Mount a disc volume and make its contents available.

**Syntax:** MC lu

lu LU number of the disc volume to be mounted. Must be a positive number.

### MO (Move Files)

**Purpose:** Move files from one directory to another, within a disc volume.

**Syntax:** MO mask1 mask2

mask1 Source file.

mask2 Destination file.

### OF (Stop or Remove Program)\*

**Purpose:** Stop a scheduled program or release a program ID segment.

**Syntax:** OF[ prog[ ID]]

prog Program name.

ID Releases the ID segment.

### OWNER (Display or Modify Owner)

**Purpose:** Display or change the owner of a directory or a subdirectory.

**Syntax:** OWNER directory[ newOwner]

directory Name of directory.

newOwner Name of new owner.

## PR (Change Program Priority)\*

**Purpose:** Change or display priority of a restored program.

**Syntax:** PR prog[ priority]

**prog** Program name.

**priority** Range is between 1 and 32767. If omitted, the priority of prog is displayed.

\*This command can be entered in response to a SYSTEM prompt.

## PROT (Display or Modify Protection)

**Purpose:** Display or change the protection status of a file or directory.

**Syntax:** PROT mask [owner/users]

**mask** File mask that includes all fields of the file descriptor and a qualifier.

**owners/** Access allowed for owner and other users  
**users** (r = read, w = write). The current protection status is displayed.

## PU (Purge Files)

**Purpose:** Purge files.

**Syntax:** PU mask[ OK]

**mask** File descriptor.

**OK** Suppresses user prompt.

## RN (Rename File or Directory)

**Purpose:** Rename a file or directory.

**Syntax:** RN mask1 mask2

**mask1** Current name.

**mask2** New name.

## COMMAND INTERPRETER (CI) COMMANDS

### RP (Restore Program File)

**Purpose:** Establish a permanent program ID segment.

**Syntax:** RP file[ prog]

**file** File name. The first five characters of the file name are used as the program name, unless the optional parameter is specified.

**prog** Program name to be used.

\*This command can be entered in response to a SYSTEM prompt.

### RU (Run Program)\*

**Purpose:** Immediately schedule a program for execution and wait for its completion.

**Syntax:** [RU ]prog|file[ param\*5]

**RU** Required only if the program name starts with two characters that can be interpreted as a CI command.

**prog|file** A five-character program name or a file descriptor.

**param\*5** Parameters to be passed to the program. The maximum run string length, including the implied RU and delimiter, is 80 characters. This can be five parameters or one long character string.

### SL [,lu]

Display linkage information for session logical unit number.

**lu** Session logical unit number (default=list information for all session lu's in user's Session Switch Table).

**SL, session lu, system lu**

Map a new session lu to system lu currently in the user's Session Switch Table. Requires capability of 30.

Add a system lu to user's Session Switch Table. Requires capability of 50.

**system lu**    May be specified as — (a dash) to delete lu mappings which have been created during user's session.

\*This command can be entered in response to a SYSTEM prompt.

**SS (Suspend Program)\***

**Purpose:**        Suspend an active program.

**Syntax:**        SS[ prog]  
                   prog        Name of an active program.

**SZ (Display or Modify Program Size)\***

**Purpose:**        Display or modify program size information.

**Syntax:**        SZ program [size[ msegSize]]  
                   prog        Program name.  
                   size        Program size in pages (for a non-VMA program) or EMA size (for an EMA program), not including PTE. Range is  $2 \leq \text{size} \leq 1022$  for EMA size.  
                   msegSize    New MSEG size for EMA programs. Range is  $1 \leq \text{MSEG size} \leq 30$ .



## COMMAND INTERPRETER (CI) COMMANDS

### TM (Display or Set System Clock)\*

**Purpose:** Display or set the system clock.

**Syntax:** TM[ month day year hr:min:sec pm]

**month** Jan to Dec

**day** 1 to 31

**year** 1976 to 2144

**hr** 0 (default) to 23

**min** 0 (default) to 59

**sec** 0 (default) to 59

**pm** AM (default) or PM

### TO (Display or Modify Device Time Out)\*

**Purpose:** Display or set time-out limit for a device.

**Syntax:** TO EQT[ interval]

**EQT** EQT number of device.

**interval** Time-out value for device LU (in 10-ms intervals).  $0 \leq \text{interval} \leq 65534$ . If interval = 0, device does not time out.

\*This command can be entered in response to a SYSTEM prompt.

### TR (Transfer to Command File)

**Purpose:** Transfer control to a command file.

**Syntax:** TR file[ param\*9]

**file** File containing CI commands.

**param\*9** One to nine parameters can be specified. They replace the variable parameters \$1 through \$9 in the command file. (Defaults to zero-length strings.) The variable parameters are described at the end of this section.

## UL (Unlock Shareable EMA Partition)\*

**Purpose:** Unlock a shareable EMA partition.

**Syntax:** UL label

**label** Identifies a shareable EMA partition label. "WH,SH" lists available labels.

## UNPU (Unpurge Files)

**Purpose:** Recover purged files.

**Syntax:** UNPU mask

**mask** File descriptor. A file can only be unpurged if its spae has not been reused. "DL,@.@.P" lists purged files.

## UP (Up a Device)\*

**Purpose:** Notify the system that a specified device is available.

**Syntax:** UP EQT

**EQT** EQT number of the device.

## VS (Display or Modify Virtual EMA Size)\*

**Purpose:** Display or change the virtual EMA size of a restored program.

**Syntax:** VS prog[ vsSize]

**prog** Program name.

**vsSize** Virtual EMA size in pages.  $32 \leq \text{vsSize} \leq 65536$ .

## WD (Display or Modify Working Directory)

**Purpose:** Display or change the working directory.

**Syntax:** WD[ directory[ file][ +s]]

**directory** Name of new working directory.

**file** Command stack file.

**+s** Post command stack to command stack file.

## COMMAND INTERPRETER (CI) COMMANDS

### WH (System Status Reporting)

**Purpose:** Report system status information.

**Syntax:** WH[,lu[,option[,program]]  
or  
WH[,option[,program]]

**lu** the session lu for display. (default=user's terminal).

**option** default User's session programs.

**AL** Display status of all suspended and scheduled programs.

**SM** Similar to AL except, state 3 programs without father son relationships are not listed.

**PA** Display status of all partitions.

**PL**

or Display all ID segments.

**PR**

**program** only display information of program (PL/PR option only).

\*This command can be entered in response to a SYSTEM prompt.

### WS (Display or Modify VMA Working Set Size)\*

**Purpose:** Display or modify VMA working set size or a restored program.

**Syntax:** WS prog[ wsSize]

**prog** Program name.

**wsSize** Working set size in pages (not including PTE).  $2 \leq \text{wsSize} \leq 1022$ .

\*This command can be entered in response to a System prompt.

## XQ (Run Program Without Wait)

**Purpose:** Schedule a program for execution, then return to CI.

**Syntax:** XQ prog|file[ param\*5]

**prog|file** Program name or file descriptor.

**param\*5** Parameters to be passed to the program. The total run string has a limit of 80 characters.

## ? (Help)

**Purpose:** Display information on a CI command.

**Syntax:** ? command.

**command** CI command.

## / (Command Stack)

**Purpose:** Display the command stack to allow selection of a previously entered command for execution. Up to 20 commands of the stack will be displayed.

**Syntax:** /[n]

**n** is a command line count that specifies the number of command lines from the last command entered. Then up to 20 of the most current commands are displayed beginning with the command line specified. The cursor is positioned at the top of the display.

A slash (/) can be used instead of a number. the number of slashes indicate how many lines to go back into the stack. For example, two slashes after the command moves two lines into the stack (CI.65> //).

## \$1-\$9 (Variable Parameters)

Nine variable parameters can be passed via the TR command to a command file. The parameters in the TR command are stored in variables 1 to 9. They are recalled by \$1 to \$9 in the command file.






# FMGR COMMANDS

CONTENT	PAGE
AC .....	C-2
AN .....	C-2
CA .....	C-2
CL .....	C-3
CN .....	C-3
CO .....	C-3
CR .....	C-4
CS .....	C-5
CT .....	C-5
DC .....	C-5
DL .....	C-6
DP .....	C-6
DU .....	C-6
EX .....	C-7
HE .....	C-7
IF .....	C-7
IN .....	C-8
LI .....	C-8
LL .....	C-8
LO .....	C-8
MC .....	C-9
ME .....	C-9
OF .....	C-9
PA .....	C-9
PK .....	C-10
PU .....	C-10
RN .....	C-10
RP .....	C-10
RT .....	C-10
RU .....	C-10
SE .....	C-11
SL .....	C-11
SM .....	C-13
SP .....	C-13
ST .....	C-13
SV .....	C-14
SY .....	C-14
TE .....	C-14
TR .....	C-14
VL .....	C-15
WH .....	C-15
?? .....	C-15
* .....	C-15
COMMAND STACKING .....	C-15

## SCHEDULING FMGR

RU,FMGR[,namr[,list[,severity code[,log]]]]

namr	File name or lu containing command input.	
log	lu of log device (default=input or LU1).	
list	lu of list device (default=LU1).	
severity code	Display commands and error codes.	
	0 Display all commands and errors (default).	
	1 Display no commands, all errors.	
	2 Display no commands, no errors except those requiring response. Terminates job on serious error.	
	3 Same as 2 except job not terminated.	
	4 Display no commands, no errors, and do not abort job.	

AC,crn[,P/G[,size[,id[,# dir. tracks]]]]

20

Allocate a cartridge to the session user from the spare cartridge pool,

crn	Cartridge reference number to be assigned to the allocated cartridge.	
P/G	Private (P) or group (G) cartridge designation (default=P).	
size	Number of tracks needed on cartridge.	
id	ASCII identifier of cartridge (default=DC00XX;XX is system lu number of terminal).	
#dir. tracks	# of tracks used by file directory (default=1).	

AN,message



20

Print message on list device.

CA,global#[,pl[opl,p2[... ,op(n),p(n+1)]]]

40

Calculate global parameter values.

global#	Integer preceding G in G-type global, or "integer:P" for P-type globals.	
pl-pn	Values used in calculations; if omitted, global is nulled.	

**opl-opn** Operations performed on operands pl-pn.

+ add two operands  
 - subtract second operand from first  
 / divide second operand by first  
 \*

, multiply two operands

O OR

X XOR (exclusive OR)

A AND

## CL[AL]

10

Display list of user accessible cartridges.

**AL** Display list of all cartridges in system.

## CN[,namr[,function[,subfnctn]]]

20

Issue control request to non-disc device.

**namr** Type 0 file name or lu (default=LU8).

**function** Control code, mnemonic (for octal see EXEC 3 call).

mnemonic

RW rewind (default=MT,CTU)

EO end-of-file

TO top-of-form (default=LP,CRT)

FF forward space file

BF backspace file

FR forward space record

BR backspace record

LE leader (default=paper tape punch)

**subfnctn** Carriage control.

+n to space n lines before next print operation.

-n page eject on line printer or space -n lines on terminal.

## CO, cartridge1, cartridge2[,options[,name1[,name2 filedes [,msc]]]]

20

Copy all or selected files from an active cartridge to active cartridge 2.

**filedes** File name, security code, and crn or mask. Minus signs (-) can be used as place holders.



## FMGR

<b>cartridge 1</b>	Source cartridge; positive crn or negative lu.
<b>cartridge 2</b>	Destination cartridge; positive crn or negative lu.
<b>options</b>	Copy options
	C clear destination cartridge
	D dump-mode
	E eliminate extents
	P purge source files after copy
	V verify
<b>name 1</b>	Starting file name.
<b>name 2</b>	Ending file name.
<b>msc</b>	Master security code.

### CR,filedes

20
----

Create a disc file — data not transferred, file subparameters required:

file type (must not be 0).  
file size (must not be 0).  
record size (when type=2).

**CR,filedes,lu,REAd,BSpace,EOf,BlNary**  
**WRite,FSpace,LEader,AScii**  
**BOTH,BOTH,PAge,cntrl**  
**,cntrl**

20
----

Create a non-disc (type 0) file — data not transferred.

<b>filedes</b>	File name, security code, and crn.
<b>lu</b>	Lu of non-disc device (positive).
<b>REAd</b>	
<b>WRite</b>	Legal input/output (no default)
<b>BOTH</b>	
<b>BSpace</b>	
<b>FSpace</b>	Legal spacing (default=FS for READ devices, no
<b>BOTH</b>	space all others).
<b>EOf</b>	
<b>LEader</b>	Control subfunction (default=EO for mass storage
<b>PAge</b>	devices, LE for paper tape punch, PA for line
<b>cntrl</b>	printer).
<b>BlNary</b>	
<b>AScii</b>	Type of data (default=AS).
<b>cntrl</b>	

**CS,lu,attribute**

30

Modify or change spool options set up by SL command.

<b>lu</b>	Lu defined at set up.
<b>attribute</b>	One of the following: <ul style="list-style-type: none"> <li>RWind reset file to first record</li> <li>PUrge change SAve flag to PUrge</li> <li>SAve change PUrge flag to SAve</li> <li>PASs remove HOld option</li> <li>END write EOF and terminate spool. Spool file placed in outspool queue (default).</li> <li>BUffer change to buffering</li> <li>NBUffer change to no buffering</li> <li>NPass change lu and/or priority information, by specifying the 2 additional parameters:           <ul style="list-style-type: none"> <li>[,outlu[,priority]]</li> <li>outlu = new lu.</li> <li>priority = new priority.</li> </ul> </li> </ul>

**CT,name[,function[,subfnctn[,message]]]**

20

Issue control request to terminal.

<b>name</b>	Type 0 file or terminal lu number.
<b>function/ subfnctn</b>	Octal code: 11B Space down a specified number of lines. subfunction: <ul style="list-style-type: none"> <li>0 skip 2 lines.</li> <li>+n skip n lines.</li> <li>-n skip n lines.</li> </ul> 20B Enable terminal (default) 21B Disable terminal 22B Set time out. Subfunction: value in units of 10 msecs.
<b>message</b>	Message to be written to terminal.

**DC,cartridge[,RR]**

10

Logically remove a cartridge from session user's environment by setting inactive bit in session control block. Non-session, deletes entry in system cartridge list.

<b>cartridge</b>	Positive cartridge reference number or negative lu.
<b>RR</b>	Session only — deletes cartridge entry in system cartridge list.

**DL** [,cartridge[,security]]  
 or  
 ,filedes[,security] 10

List the file directory of one or all of the mounted cartridges.

- cartridge** Cartridge reference number, positive for label or negative for lu. Zero or none specified lists all.
- filedes** Mask specifying the file entries in the directory to be output. Minus signs (-) can be used as place holders for more flexibility.
- security** Two-character FMP master security code.

If the master security code is 0, default in command will not obtain long list showing security codes — a code (any code) must be supplied.

**DP** [,p1[,p2[,p3...[,pn]]]] 20

Display parameter value or global names. p1-pn are parameters to be displayed.

**DU**,namr1,namr2[,record format[,file#[,#files]]] 20

Transfer data from an existing file or lu to another existing file or lu. Does not create namr2.

- namr1** Source of data
- namr2** Destination of data
- record format** Format of data or EOF control (default=namr1 format, or ASCII if non-disc device).
  - ASCII ASCII records.
  - BReloc Binary relocatable records with checksum.
  - BNary Binary records without checksum.
  - BAbs Binary absolute records with checksum.
  - MTape Magnetic tape ASCII records.
  - MS Magnetic tape SIO (System Input/Output) records are written on namr2. Standard records are expected on namr1.
  - MSBR Magnetic tape SIO binary relocatable records (same as MS+BR).
  - MSBA Magnetic tape SIO binary absolute records (same as MS+BA).
  - IHibit Inhibits EOF on namr2 and leader punching.
  - SAve Save embedded EOF's in namr1.

**file#** File or subfile on namr2 where transfer starts (default=1).

**#files** Number of files to be transferred from namr1 (default=1).

**EX**

Terminate FMGR.

1

**EX, SP**  
**RP** [,RG[,KI]]

Initiate log-off process.

**SP/RP** Save/release private cartridges.

**RG** Release group cartridges.

**KI** Abort any active session programs.

1

**HE[,keyword[,lu]]**

Detailed error code explanation.

**keyword** Identifiers related to error code (session default=last error posted). Non-session, keyword must be specified.

**lu** Device for explanation output (default=user's terminal).

1

**IF,p1,xx,p2[,skip]**

Compare two values (usually globals) and skip a specified number of commands. Command not allowed from interactive device, must be in procedure file or batch job.

**p1,p2** Values to be compared.

**xx** ASCII operators as follows:

EQ  $p1 = p2$

NE  $p1 \neq p2$

LT  $p1 < p2$

GT  $p1 > p2$

GE  $p1 \geq p2$

LE  $p1 \leq p2$

**skip** Number of commands to skip (positive or negative). Use -2 to skip back to previous command (default=1).

40

**IN,mstr sec code,ctrldge,lbl,id[,1st trk[,#dir trks[,#sec/trk[,bad trks]]]]**

60

Initialize a cartridge.

- mstr sec code** Ignored in a session environment.
- ctrldge** Cartridge reference number, positive for label or negative lu. (Must be -lu if new.)
- lbl** New cartridge reference label; 0 < lbl < 32768, or two ASCII characters.
- id** Cartridge information label; up to 6 ASCII characters, must follow FMGR namr restrictions.
- 1st trk** First track to be used on the cartridge. If LU2, must be 8 greater than last system track (default=track 0).
- #dir trks** Number of directory tracks (1 to 48), (default=1).
- #sec/trk** Number of 64-word sectors per track. If LU2/3, parameter is ignored.
- bad trks** Bad track list. Up to six track numbers separated by commas.

**IN,master security code -- new security code**

60

Change master security code. New code is separated from old code by two minus (-) signs.

**LI,namr[,format[,ln1[,ln2]]]**

10

List contents of a file or lu on list device.

- format** Specifies list format.
  - S Source (default for type 0,3,4 files).
  - B Binary (default for all other type files).
  - D Directory information only.
- ln1** Starting line.
- ln2** Ending line.

**LL,namr**

20

Change current assignment of list device, namr may be either file or lu number.

**LO,lu**

40

Change lu number of log device where lu is an interactive device.

**MC,lu[,P/G[,size[,id[,#dir trks[,label]]]]]**

10

Make an unmounted cartridge available for use.

<b>lu</b>	Lu number of cartridge to be mounted, it must be in user's session switch table.
<b>P/G</b>	Private or group cartridge (session default=P) non-session meaningless, but its space must be provided.
<b>size</b>	# of tracks needed on cartridge.
<b>id</b>	ASCII identifier of cartridge (default DC00XX; XX is system lu number of terminal).
<b>#dir trks</b>	# of tracks used by the file directory (default=1).
<b>label</b>	Cartridge reference number to be assigned to the cartridge.

**ME[,namr[,clear]]**

10

Display contents of user's message file.

<b>namr</b>	File name or non-disc lu to receive messages (default=user's terminal).
<b>clear</b>	1 (clear message file). 0 (do not clear=default).

**OF,program**

30

Terminate program within caller's current session.

**OF,program**

60

Terminate any program within the system.

**PA[,lu[,message]]**

40

Suspend execution of the current job or procedure file, and transfer control to a specified device, and optionally print a message.

<b>lu</b>	Lu to which control transfers (default=log device).
<b>message</b>	1-80 ASCII characters.

**PK[,cartridge]** 20

Recover tracks and directory entries assigned to purged files and close gaps between files.

**cartridge** Cartridge reference number, positive for label or negative for lu (default=all user accessible cartridges).



**PU,filedes** 20

Remove a file and its extents from system.



**RN,filedes,newdes** 20

Change a file name and attributes.

**filedes** Existing file name and parameters.

**newdes** New name, file type, and/or security code. No other subparameters may be changed.



**RP,filedes,program[,pname]** 30

Restore program file "filedes" using the ID segment of "program", renaming the restored program to pname.

**RP,filedes[, ,pname]** 30

Restore program file "filedes", which must be a type 6 file on any cartridge, renaming the restored program to pname.

**RP,,program** 30

Release "program's" ID segment where "program" is a program with its ID segment in memory.



**RT,program** 30

Release all disc tracks assigned to a dormant program.

**RU,program:IH[,parameters]** 30

Schedule "program" for immediate execution, inhibit automatic renaming feature.



**RU**[IH],program[,parameters]

30

Schedule "program" for immediate execution. IH inhibits passing of command string.

**program** Name of program to be executed or name of type 6 file containing program or procedure file to be executed.

**parameters** 1-5 parameters to be passed to program or 1-9 parameters passed to a procedure file.

**SE**[,p1[,p2[,...[p9]]]]

40

Set or clear global parameters 1G-9G where p1-p9 are values to be converted to global parameters. If all parameters omitted, globals are nulled. If any one parameter omitted, corresponding global unchanged.

**SL**[,lu]

10

Display linkage information for session logical unit number.

**lu** Session logical unit number (default=list information for all session lu's in user's Session Switch Table).

**SL**,lu[,filedes[,attribute[,outlu[,priority[,prog]]]]]

30/50

Spool setup and outspool control.

**lu** The session lu to which a spool file is to be associated. The lu must not be LU2 (system disc), LU3 (auxiliary disc), any lu associated with a disc driver, a spool lu, or if in a job system LU5 (standard spool input device).

**filedes** Name of existing file to be used as a spool file (default=system assigns spool pool file).

**attribute** Defines characteristics of spool access. Any 3 attribute codes can be combined, no delimiters necessary.



attribute codes:

- NO = Queue file for immediate outspool.
- RE = Read only.
- WR = Write only.
- BO = Both read and write.
- WN = Write now.
- BU = Buffered.
- PU = Purge.
- SH = Write spool headers.
- ST = Standard file format.

default for attribute codes:

	filedes not specified	filedes specified
outlu specified	WR HO SH SP	WR HO SH SA
outlu not specified	BO HO ST SP	RE HO ST SA

- SP = Spool pool file
- SA = Save (don't purge)
- HO = Hold till close

- priority**      Outspool priority (default=session — 99, batch — priority of job).
- prog**            If specified, program "prog" will be scheduled, with wait, by the spool system when spool lu is closed. Note the spool file will not be outspooled, "prog" must properly dispose of the file. Requires capability of 50.
- outlu**            Session lu for outspooling.

**SL,session lu,system lu**

30/50

- Map a new session lu to system lu currently in the user's Session Switch Table. Requires capability of 30.
- Add a system lu to user's Session Switch Table. Requires capability of 50.
- system lu**      May be specified as — (a dash) to delete lu mappings which have been created during user's session.

**SM,user,namr,message**

10

Send message and/or file to another user's message file.

<b>user</b>	Log on ID of message recipient, (user.group).
<b>namr</b>	Name of file or non-disc lu containing data to be sent.
<b>message</b>	String entered from sender's terminal.

**SP,filedes[,PR  
or [,capability]]  
,GR**

30

Place a disc resident program and its ID segment in a type 6 file created by this command. First 5 characters of file name must be identical to disc program name. File subparameters default to:

security	0
cartridge	first cartridge in user's cartridge list
file type	type 6
file size	size of program
record size	128

**ST,namr1,namr2[,record format[,eof]  
[,file#[,#files]]]**

20

Transfer data from an existing file or lu to another file or lu. namr2 created by this command.

<b>namr1</b>	Source of data.
<b>namr2</b>	Destination of data.
<b>record format</b>	Format of data or EOF control (default=namr1 format or ASCII if non-disc device).
ASCII	ASCII records.
BReloc	Binatable records with checksum.
BNary	Binary records without checksum.
BAbs	Binary absolute records with checksum.
MTape	Magnetic tape ASCII records.
MS	Magnetic tape SIO (System Input/Output) records are expected on namr1. Standard records are written on namr2.
MSBR	Magnetic tape SIO binary relocatable records (same as MS+BR).

## FMGR

- eof** EOf control.  
IHibit Inhibits EOF on namr2 and leader punching.  
SAve Save embedded EOF's in namr1.
- file #** File or subfile on namr1 where transfer starts (default=1).
- #files** Number of files to be transferred from namr1 (default=1).

### SV,severity[,global #][,IH]

20

Change the system log device severity code to a new number.

- severity** 0 display all commands and errors (default).  
1 display no commands, all errors.  
2 display no commands, no errors except those requiring response. A serious error terminates job.  
3 display same as 2, except job not terminated.  
4 display no commands, no errors, job not terminated.
- global #** Optional G global number (1-9) into which current severity code is to be placed.
- IH** Optional parameter to inhibit echo of command entry.

### SYcommand

1

Execute RTE system command from FMGR.  
Preface command by SY (use no delimiter, e.g., SYTI).

### TE,message

10

Send message to the operator via the system console.

### TR[,xfer[,parameters]]

1

Transfer control to a file or lu, passing parameters as globals.  
A comma (,) or colon (:) may replace the "TR," as the transfer command code.

- xfer** A negative integer that denotes a transfer back that many files, or the name of a file or lu.
- parameters** The parameters to be set into the globals (1G-9G). Skipped parameters are not changed.

**VL,cartridge**

60

Assign system scratch and VMA backing store cartridge.

**cartridge** Positive cartridge reference number or negative lu; default (command not entered or cartridge = 0) is first available cartridge in user's cartridge list.

**WH[,lu[,option[,program]]]**

10

or

**WH[,option[,program]]**

Schedule WHZAT program.

**lu** The session lu for display.

**option** default User's session programs.

AL Display status of all the suspended and scheduled programs.

SM Similar to AL except state 3 programs without father son relationships are not listed.

PA Display status of all partitions.

PL

or Display all ID segments.

PR

**program** Only display information of program (PL/PR option only).

**??[error#]**

10

Request FMGR error code explanation.

**error#** FMGR error code (default=last error issued).

**\*COMMENT LINE**

10

**COMMAND STACKING**

**Ln** "n" is the number of lines to list (default is to list the entire command stack).

**P** Display or edit the pending line in the command stack. Edit options are CNTL/R, CNTL/I, CNTL/S, CNTL/T and CNTL/C. See the Chapter on the Interactive Editor.

## FMGR

- n Position pending line to the "n"th line in the command stack.
- ^n or Rn Position "n" lines preceding pending line.
- /n Position "n" lines past pending line.
- n Delete "n" lines from command stack from the pending line.

Once a line has been displayed as the pending line, it may be executed by typing a carriage return.

# SYSTEM AND BREAKMODE COMMANDS

CONTENT	PAGE
AB .....	D-2
AG* .....	D-2
AS* .....	D-2
BL* .....	D-2
BR* .....	D-2
CU* .....	D-2
DN* .....	D-2
EN .....	D-3
EQ* .....	D-3
FL .....	D-3
GO .....	D-4
HE* .....	D-4
IT* .....	D-4
LU* .....	D-4
OF* .....	D-5
ON* .....	D-5
OP .....	D-5
PR* .....	D-5
QU* .....	D-5
RS .....	D-6
RT .....	D-6
RU* .....	D-6
SL* .....	D-6
SS* .....	D-6
ST* .....	D-6
SZ* .....	D-7
TE .....	D-7
TI* .....	D-7
TM* .....	D-8
TO* .....	D-8
UL* .....	D-8
UP* .....	D-8
UR* .....	D-8
VS* .....	D-8
WH* .....	D-9
WS* .....	D-9

\*These commands are also available in CI



**DN,eqt\***

60

Set I/O controller down.

**eqt** equipment table entry number.

**EN,mstr scty code[,option]**

--

Enable system console as a session terminal. Command only valid when entered from the system console.

**mstr scty code** Two character FMP master security code.

**option** 0 master security code not required in "OP" commands (default).

1 master security code is required in "OP" commands.

**EQ,eqt\***

10

Print description and status of an I/O controller. Status information is printed as.

**select code DV.nn D B Unnn status**

**select code** is the I/O select code number.

**DV.nn** is the driver routine.

**D** is D if DMA required; 0 if not.

**B** is B if automatic output buffering; 0 if not.

**Unnn** is the last subchannel addressed.

**status** is the logical status:  
 0 = available.  
 1 = I/O controller down.  
 2 = I/O controller busy.  
 3 = waiting for DMA assignment.

**EQ,eqt, UNbuffer \*  
BUffer**

60

Change the automatic buffering designation for a particular I/O device.

**FL**

10

Eliminate buffered output to a session terminal. Only valid in break mode, and not valid from system console.



## SYSTEM AND BREAKMODE

**GO**[IH][,program][,pl[,...[,p5]]]]

30/60

Reschedule any program in users session, where parameters are passed by program only when it has suspended itself. GOIH inhibits passing of command string. Requires capability of 30.

Reschedule any program in the system. Requires capability of 60.

**HE**[,keyword[,lu]]\*

1

Detailed error explanation.

**keyword** an eight character error iode (default=last error logged).

**lu** device for explanation (default=user's terminal).

**IT**,program[,res,mpt[,hr,min[,sec[,tms]]]]\*

50

Set automatic execution time value for a program. ON command must follow to schedule the program. Not specifying optional parameters removes "program" from the timelist (program must be dormant).

**res** resolution code:

1 tens of ms

2 seconds

3 minutes

4 hours

**mpt** multiplier (0-4095) used with res.

**hr,min** Initial start time.

**sec,tms**

**LU**,lu\*

60

Print EQT entry number, device subchannel number, associated with a system lu, and whether the device is up or down. See SL command for similar function.

**LU**,lu,0\*

60

Reassign system lu to be bit bucket.

**LU**,lu,eqt[,subchannel #]\*

60

Reassign new EQT entry number to system lu. If EQT number has subchannels, use subchannel #.

**OF,program[,numb[,NP]]\***

30/60

Terminate a session program. Requires capability of 30.

Terminate any program in the system. Requires capability of 60.

- numb**      0 remove from time list; disc tracks not released (default).  
               1 terminate immediately; release disc tracks  
               ID or 8 terminate immediately and permanently from system (must be issued to segments as well as the main).
- NP**            abort message is not printed.

**ON[IH],program[,NOW][,parameters]\***

50

Schedule a program for execution. Program's entry in time list is affected. ONIH inhibits passing of command string.

- NOW**          Schedule program immediately.
- parameters** 1-5 parameters passed to program when it is scheduled.

**OP[,mstr scty code[,command]]**

--

Enter a system level command from a low capability session. Command only valid when entered from the system console.

- mstr scty code** Two character FMP master security code. If specified in the "EN" command the security code is required.
- command**      The system command to be executed.

**PR,program,priority\***

50

Change program priority where priority = 1-32767 (decimal).

**QU[,quantum[,limit]]\***

10/60

Examine system timeslice quantum and fence. Requires capability of 10.

Modify system timeslice quantum and fence. Requires capability of 60.

- quantum**      system timeslice quantum, value 0-32767 millisecs (default=1500).
- limit**         priority level fence to begin timeslicing (default=50).

## SYSTEM AND BREAKMODE

**RS**

10

Abort and reschedule a session's copy of FMGR or CI.

**RT,program**

30

Release all disc tracks assigned to a program.

**RU[IH],program[,parameters]\***

30

Schedule a program for immediate execution. Program's entry in time list is not affected. 1-5 parameters are optionally passed to program when it is scheduled. RUIH inhibits saving of command string. The breakmode RU actually runs "program" not a renamed copy of "program"

**SL[,lu]\***

10

Display session lu information.

lu session lu for which linkage information is desired.  
(Default=information for all session lu's in user's session switch table.)

**SS[,program]\***

30/60

Suspend non-dormant session program. Requires capability of 30. If program name not specified, the current session program is suspended.

Suspend non-dormant system program. Requires capability of 60.

**ST,name\***

10

Determine status of named program. Status is printed as:

pr S res mpt hr min sec ms T

pr Decimal priority.

S current state of program:

0 Dormant

1 Scheduled

2 I/O suspend

3 General wait

4 Unavailable memory suspend

5 Disc allocation suspend

6 SS or EXEC 7 suspend

9 Background segment

res/mpt/ 0 or  
hr/min/sec time program is next scheduled to run.  
/ms

T Program currently in time list.

### ST[,numb]\*

10

Determine name or partition number of program currently executing.

numb 0 — Display name and partition number of program currently executing in memory. 0 displayed if none executing.

Partition # — Display name of program currently residing in that partition. 0 if none.

### SZ,program\*

30

Display the named program's size information as follows:

AAAAA BB CCCC DDDD EE

AAAAA last word plus 1 of program.

BB required program size (includes MSEG if necessary).

CCCC required partition size. Program code + EMA.

DDDD EMA size (EMA programs only).

EE MSEG size (EMA programs only).

### SZ,program,size[,MSEG size]\*

30

Change size of "program".

program program name.

size Non-EMA program: required program size.  
EMA program: required EMA size.

MSEG size new MSEG size (EMA program only).

### TE,message

10

Send message to system console.

### TI\*

10

Print current year, Julian day and time.

## SYSTEM AND BREAKMODE

**TM,year,day[,hr[,min[,sec]]]\***

60

Set real time clock.

year four digits (e.g., 1957).

day three digits Julian date (e.g., 063 = March 4).

**TO,eqt[,numb]\***

10/60

Examine device time out parameters. Requires capability of 10.

Change device time out parameters. Where numb is number of 10 ms intervals used as new time out value. Requires capability of 60.

**UL,label\***

60

Unlock a shareable EMA partition.

**UP,eqt\***

10

Make I/O controller (and all associated lu's) available.

**UR,partition #\***

50

Release reserved partition.

**VS,program\***

30

Display the virtual memory size of program as follows:

AAAAA BB CCCC DDDD EE FFFFF

AAAAA last word plus 1 of program

BB required program size (pages)

CCCC required partition size. (Program code and working set)

DDDD working set size (pages)

EE MSEG size

FFFFF last page of virtual memory

**VS,program,lastpg\***

30

Modify the virtual memory size of program to lastpg+1 (pages).

lastpg request last page of virtual memory size (default=8191 pages).

**WH**[,lu[,option[,program]]]\*

or

**WH**[,option[,program]]\*

10

Schedule WHZAT program.

<b>lu</b>	the session lu for display. (default=user's terminal).
<b>option</b>	default User's session programs.
	AL Display status of all suspended and scheduled programs.
	SM Similar to AL except, state 3 programs without father son relationships are not listed.
	PA Display status of all partitions.
	PL
	or Display all ID segments.
	PR
<b>program</b>	only display information of program (PL/PR option only).

**WS,program\***

30

Display the working set size of program (pages).

See the VS command for an explanation of the output.

**WS,program,wrksz\***

30

Modify working set size of program

<b>wrksz</b>	new working set size of program (default=31 pages).
--------------	---




# EDIT/1000 COMMANDS

CONTENT	PAGE
RU,EDIT .....	E-2
POSSIBLE USER RESPONSE .....	E-2
PARAMETER CONVENTIONS .....	E-2
OPTION SETTING COMMANDS .....	E-2
CONTROL COMMANDS .....	E-3
SCREEN EDIT COMMANDS .....	E-3
DISPLAY COMMANDS .....	E-4
LINE EDITS .....	E-5
CHARACTER EDITS .....	E-5
SEARCH COMMANDS .....	E-5
EXCHANGE COMMANDS .....	E-6
FILE INPUT/OUTPUT COMMANDS .....	E-6
TERMINATIONS .....	E-6
COMMAND STACK .....	E-6




**RU,EDIT[filedes[,commands]]**

**filedes** File to be edited.

**commands** Edit commands to be executed upon entering EDIT. Commands are separated by a '|'. 

**POSSIBLE USER RESPONSE**


{ } (blank). Current area copies to EDIT's work area.

A Abort EDIT immediately. 

F|,filedes File to be copied to EDIT's work area.

EDIT prompt character '/' (default).

**PARAMETER CONVENTIONS**

Current line. 

\$ or ) Last line in file.

n Line number.

lr Line range (absolute line numbers or an offset from the current line).

**OPTION SETTING COMMANDS**

SE,option Set various EDIT options and defaults.

EDIT options

AC — anchor character (default=^)

EC — escape character (default=\\)

IC — indefinite character (default=@)

PC — prompt character (default=|)

CS — command separator (default=|)


TC — tab character (default=TAB key or CNTL/I)

WC — window columns (default=1,150)

SD — screen size (default=10,10,2)

SL — screen length (default=30 or 100)

VW — vertical window (default=10,10)


LE — maximum characters on line (default=150) 

AS — verify dangerous commands (default=on)

CF — case folding (default=on)

RE — regular expression (default=off)

DF — display function (default=on)

RT — return to pending line after multiple search (default=on) 

TS — time stamp update (default=on)

BE — bell with prompt (default=off)

T[n1,...,n10]	Set tab columns up to 10 settings.
TA	Set tab columns to 7 and 21.
TF	Set tab columns for FORTRAN (7, then every 4).
TL	Set terminal tab stops to line mode tabs.
TM	Set tab columns for Macro (10,26,40,44,48).
TP	Set tab columns for Pascal (every 3 columns).
TS	Set terminal tab stops to screen mode tabs.

## CONTROL COMMANDS

BK	Kill trailing blanks.
RU,program	Run program and return to EDIT at pending line upon termination.
SC	Copy screen memory after pending line.
TR,namr, [Q]	Transfer to a command file or an input device. Specify Q to suppress the listing.
UN	Revert the command executed immediately before this command to the prior state.
UY	Copy lines from undo list into work file.

## SCREEN EDIT COMMANDS

[lr]S Start screen edit.

### SCREEN MODE COMMANDS

CNTL/Q	Quit screen mode.
CNTL/P	Go to previous screen.
CNTL/F	Forward one screen.
CNTL/S	Start next screen.
CNTL/X	Start next screen with larger screen size.
CNTL/C	Execute command and return to screen mode.
CNTL/O	Copy line on the screen.
CNTL/A	Move to first non-blank character on line.
CNTL/Z	Move to last non-blank character on line.
CNTL/K	Set line market at current line.

Screen mode commands entered twice leave the screen unchanged.

## DISPLAY COMMANDS

?[command] Display information about the specified command  
or (default=summary of commands).

H[command]

?[option] Display the requested information.  
or

H[option]

option

EX = explanation of abbreviations.

LS = line specification explanation.

PA = pattern explanation (? only).

PL = pending line edit information (? only).

RE = regular expression explanation.

HL Display header lines.

[lr]L[max] Display a maximum number of lines (default=20).

[+][list file] The lines can be listed or appended to a list file.

LE Display the number of characters in the pending line.

LI Display the number of lines in the file.

n Display line n, make it the pending line.

+n Forward n lines and display new pending line.

-n Go back n lines and display new pending line.

N Display the line number of the pending line.

SH[option] Display the specified EDIT option  
(default=summary of all EDIT options).

SZ Display approximate number of words in the destination file.

[lr]W List a vertical window.

## LINE EDITS

- [n] I<text> Insert text before specified line (default=pending line).
- { }text Add text after pending line.
- [lr]K[max] Delete the specified lines (default=pending line).  
[+][list namr] The deleted lines can be saved or appended to a list file.
- C<text> Edit pending line then advance pending line.
- [n]O<text> Duplicate the specified line and edit (default= pending line.)
- [n]P<text> Edit the specified line and display (default=pending line).
- Q Edit pending line with terminal edit keys.
- [n]R<text> Replace the specified line with text (default= pending line).
- [n]J Join the specified line to the following line (default=pending line).

## CHARACTER EDITS

- CNTL/B Break line at cursor position.
- CNTL/C Delete characters.
- CNTL/R Replace characters.
- CNTL/S Insert characters.
- CNTL/T Truncate line at cursor position.
- CNTL/X Extend line.

## SEARCH COMMANDS

- [lr]B/pattern/ Find a line with "pattern" within the specified line range [A,N,Q,V] (default=SOF to EOF).
- [lr]F/pattern/ Find a line with "pattern" within the specified line range [A,N,Q,V] (default=after pending line to EOF).
- [lr]D/pattern/ Delete lines from the specified line to the line containing "pattern" (default=pending line to EOF).

**EXCHANGE COMMANDS**

- [lr]G/old/      Exchange old with new over the specified range  
new/[N,R,S] (default=pending line).
- [n]Y/old/      Exchange old with new on the specified line and  
new/            display the next occurrence (default=pending line).  
[N,Q,R,S]
- [lr]X/old/      Exchange old with new over the specified range  
new/            (default=pending line).  
[N,Q,R,S]
- [lr]U/old/      Unconditional character replace over the specified  
new/[Q]        range.

**OPTIONS**

- A    — Multiple search
- N    — No-window parameter
- Q    — Display suppression
- V    — Reverse match
- R    — Remove zero length records
- S    — Single exchange parameter

**FILE INPUT/OUTPUT COMMANDS**

- FCL            Close the list file (opened with the K or L command).
- FCS            Close the source file.
- FI filedes    Replace the source file with another file.
- M filedes    Merge file after pending line.
- WC filedes   Create file without exiting EDIT.
- WR filedes   Replace file without exiting EDIT.

**TERMINATIONS**

- A            Abort leaving source file unchanged.
- EC filedes   Create a FMGR file and end EDIT session.
- ER           Replace source file and end EDIT session.
- ER filedes   Replace specified file and end EDIT session.

**COMMAND STACK**

- /[n]        Display the command stack to allow selection of a  
previously entered command for execution. Up to 20  
commands of the stack will be displayed.

## LINK COMMANDS

CONTENT	PAGE
LINK RUN STRING .....	F-2
AB (ABORT) .....	F-2
AS (ASSIGN PARTITION) .....	F-2
CR (SPECIFY SCRATCH VOLUME — FMGR ONLY) .....	F-2
DB (DBUGR) .....	F-2
DE (DEBUG) .....	F-3
DI (DISPLAY) .....	F-3
DP (DO NOT PURGE) .....	F-3
EC (ECHO) .....	F-3
EM (EXTENDED MEMORY ACCESS) .....	F-3
EN (END) .....	F-3
FO (FORCED LOAD) .....	F-4
LC (LABELED SYSTEM COMMON) .....	F-4
LI (LIBRARY) .....	F-4
LK (RELINK) .....	F-4
LL (LIST OPTION) .....	F-5
MA (LOAD DISPLAY MAP) .....	F-5
OS (OPERATOR SUSPEND) .....	F-5
OU (OUTPUT) .....	F-5
PR (PRIORITY) .....	F-5
PS (PAGE ALIGN OVERLAYS) .....	F-6
RE (RELOCATE) .....	F-6
RO (REORDER) .....	F-6
SC (SYSTEM COMMON) .....	F-6
SE (SEARCH) .....	F-6
SH (SHAREABLE EMA) .....	F-7
SZ (SIZE) .....	F-7
SN (SNAPSHOT) .....	F-7
VM (VIRTUAL MEMORY) .....	F-7
VS (VIRTUAL MEMORY SIZE) .....	F-7
WS (WORKING SET SIZE OF VMA) .....	F-8
* (COMMENT) .....	F-8

## LINK COMMANDS

### LINK Run String

Syntax: LINK [param\*n]

**param\*n** Any number of parameters up to the 80-character run string limit. Commands and/or file names can be specified in any order. A command is prefixed with plus (+). Command parameters are delimited with colon (:).

If no parameters are specified in run string, LINK runs interactively.

### AB (Abort)

Purpose: Abort LINK immediately.

Syntax: AB

### AS (Assign Partition)

Purpose: Assign a partition where the program will reside.

Syntax: AS partNum

**partNum** Decimal number between 0 (default) and 1023. A partNum of 0 cancels number previously specified.

### CR (Specify Scratch Volume — FMGR Only)

Purpose: Specify volume (crn) to be used for LINK scratch files when running LINK from FMGR. Default is working directory.

Syntax: +CR:crn (in run string only)

### DB (DEBUGR)

Purpose: Append DEBUGR to program.

Syntax: DB

**DE (DEBUG)**

**Purpose:** Prepares LINK for use with the Symbolic Debug/1000 Program.

**Syntax:** DE (+DE in run string)

**DI (Display)**

**Purpose:** Display undefined externals.

**Syntax:** DI

**DP (Do Not Purge)**

**Purpose:** Inhibit purging of existing program files when running LINK interactively.

**Syntax:** +DP (in run string only)

**EC (Echo)**

**Purpose:** Echo loader command file commands.

**Syntax:** EC (+EC in run string)

**EM (Extended Memory Access)**

**Syntax:** EM size

size                      Number of pages of EMA space, in range 2 to 1022.

**EN (End)**

**Purpose:** End command input.

**Syntax:** EN [file]

file                      File descriptor of program output file. Can be defaulted to file descriptor specified by OU command, source PROGRAM statement, or .REL file name.



## LINK COMMANDS

### FO (Force Load)

**Purpose:** Force-load program or overlay, regardless of undefined externals.

**Syntax:** FO

### LC (Labeled System Common)

**Purpose:** Specify that references to labeled system common by the program will be satisfied.

**Syntax:** LC (+LC in run string)

### LI (Library)

**Purpose:** Define library file that the loader searches immediately before searching snapshot file and system libraries. Up to 10 library files can be defined by repeating this command.

**Syntax:** LI file

**file** File descriptor of file to be searched.

### LK (Relink)

**Purpose:** Change attributes of previously linked program. (LO command lists current program attributes.)

**Syntax:** LK file

**file** File descriptor of program file to be relinked.

## LL (List Option)

**Purpose:** Specify destination for list output.

**Syntax:** LL file|lu

+LL:file (in run string)

**file** File descriptor of list file. If file descriptor is specified, file must not exist or must have type extension MAP.

**lu** LU number of list device.

## MA (Load Display Map)

**Purpose:** Display load map on terminal when running LINK interactively.

**Syntax:** +MA (in run string only)

## OS (Operator Suspend)

**Purpose:** Suspend LINK.

**Syntax:** OS

## OU (Output)

**Purpose:** Specify program output file name. (LINK does not overwrite existing file by same name.)

**Syntax:** OU file

**file** File descriptor and subparameters for program output file.

## PR (Priority)

**Purpose:** Set the priority of the program.

**Syntax:** PR nn

**nn** Program priority, in range 1 to 32767 (default is 99).

## LINK COMMANDS

### PS (Page Align Overlays)

**Purpose:** Start overlays at page boundaries.

**Syntax:** PS



### RE (Relocate)

**Purpose:** Include a relocatable file as part of current segment.

**Syntax:** RE file

**file** File descriptor of file to be included in program.



### RO (Reorder)

**Purpose:** Rearrange program modules to reduce number of base page links. For CDS program, this command rearranges only non-CDS code assigned to the data segment.

**Syntax:** RO (+RO in run string)

### SC (System Common)

**Purpose:** Specify that blank common referenced by program will be placed in blank system common.

**Syntax:** SC (+SC in run string)



### SE (Search)

**Purpose:** Search a library file to satisfy undefined external references.

**Syntax:** SE [file]

**file** File descriptor of library file. Default search is through snapshot and system library files.



**SH (Shareable EMA)**

**Purpose:** Specify that the declared EMA resides in the shareable EMA partition specified.

**Syntax:** SH label.

label                    Partition name (up to 6 characters).

**SN (Snapshot)**

**Purpose:** Define or display snapshot file. (Default snapshot file name is SNAP unless changed by this command.)

**Syntax:** SN [file]

file                    File descriptor of snapshot file. If this parameter is omitted, the snapshot file name is displayed.

**SZ (Size)**

**Purpose:** Specify number of physical memory pages required to run program.

**Syntax:** SZ [nn] (+SZ:nn in run string)

nn                    Number of pages, in range 2 to 32.  
Default is current program size.

**VM (Virtual Memory)**

**Purpose:** Specify that program uses VMA.

**Syntax:** VM

**VS (Virtual Memory Size)**

**Purpose:** Specify virtual memory space size.

**Syntax:** VS size

size                    Number of pages, in range 32 to 65536  
(default is 8192).

## LINK COMMANDS

### WS (Working Set Size of VMA)

**Purpose:** Specify working set size of VMA, excluding one-page PTE.



**Syntax:** WS size

size                      Number of pages, in range 2 to 1022  
(default is 32).



### \* (Comment)

**Purpose:** Ignore remainder of line.

**Syntax:** \* (in column one)



# INTERACTIVE UTILITIES

CONTENT	PAGE
MLLDR/LOADR OPERATION .....	G-2
MLLDR/LOADR COMMANDS .....	G-3
MLLDR SEGMENTATION COMMANDS .....	G-4
SGMTR .....	G-5
SXREF .....	G-5
INDXR .....	G-5
CMD .....	G-5
GENIX .....	G-6
SCOM .....	G-6
FC .....	G-6
WRITT .....	G-7
READT .....	G-8
FORTRAN AND MACROASSEMBLER .....	G-8
PASCAL .....	G-10
COMPL AND CLOAD .....	G-10
DBUGR AND MLSDB .....	G-11
BREAKPOINT AND PROGRAM CONTROL .....	G-11
MEMORY EXAMINATION AND MODIFICATION .....	G-12
SPECIAL REGISTERS .....	G-12
MAP EXAMINATION SPECIAL MODE .....	G-13
PRINT MODE CONTROL .....	G-13
SYSTEM CONFIGURATION DISPLAY (LUPRN) .....	G-14
FOWN .....	G-14
FPACK .....	G-14
FREES .....	G-14
FSCON .....	G-15
FVERI .....	G-15
LINDX .....	G-15
MERGE .....	G-15
OLDRE .....	G-15
TF .....	G-16
FLAG .....	G-17
EXT .....	G-18
FPORT .....	G-19

## UTILITIES

### MLLDR/LOADR OPERATION

#### LOADR

RU, [command[,input[,list[,opcode

MLLDR [format[,partn[,size[,profile]]]]]

**command** A command file namr, or input device lu (default=user's terminal or LU5 if batch).

**input** The file name of the relocatable main program or the lu of the relocatable input (no default).

**list** List lu, or file name namr. If a file name is specified, the file must not already exist unless its name begins with ('). (default=user's terminal or LU5 if batch).

**opcode** Default = BGNCTE (LOADR) or EBNCTE (MLLDR)

BG Background program

RT Real time program

LB Large background program

EB Extended background program

SC System COMMON

RC Reverse COMMON

NC No COMMON

SS Use subsystem global (SSGA)

PE Permanent program

TE Temporary program

RP Replace permanent program (do not also specify PE)

VM Virtual memory program with default sizes (VMA size=8191 pages, Working set=31 pages)

EM EMA program (used in MLS programs)

**format** DB Append DBUGR (LOADR) or MLSDB (MLLDR) subroutine to the program.

LE List entry points and base page links.

NL No listing desired.

DC Don't copy. Multiple copies of the program are not desired.

MP Use current page links, except for external references.

CP Use current page links, including external references.

BP Use base page links only (default).

**partn** The specific partition number in which program is to be executed.

- size** Allows a logical address space larger than the program size. Permits use of a dynamic buffer at the end of the program.
- profile** LU for profile output. Default = no profile is done. (MLLDR only).

## MLLDR/LOADR COMMANDS

- SE** Search the system disc library for undefined externals.
- SE,file** Search specified file for undefined externals.
- MS,file** Search specified file for undefined externals. The file is searched multiple times to satisfy backward references.
- RE,file** Load specified file, which may be a program, sub-routine, or segment.
- LO,XXXXXB** Change the load address of the next module to be or, +n relocated to the specified address or offset n pages.
- LI,YYYY** Set up file YYYY as a library file. Up to 10 files may be specified.
- SL** Search all files specified in the library command.
- TR,file** Go to file for succeeding loader commands.
- TR** Return to command file suspended when the undefined external was encountered.
- FO** Force load a program or segment.
- DI** Print list of undefined externals.
- EC** Echo input commands on list device.‡
- EN**
- EX** End of command input.
- /E**
- AB** Abort the loader immediately.
- /A**
- \*** Comment line.
- AS,XX** Assign the relocated program to partition XX.‡
- SZ,YY** Allows a logical address space larger than the program size. Permits the use of a dynamic buffer at the end of the program.‡
- or, +n**



## UTILITIES

LL,namr	Lu or file name for listing. If a file it must not already exist, unless its name begins with (').‡	
OP,opcode	Specify an opcode parameter. See opcode section of MLLDR/LOADR OPERATION.‡	☾
FM,format	Specify a format parameter. See format section of MLLDR/LOADR OPERATION.‡	
PF,lu	Append profiling subroutine to the program. List profile output to lu. (MLLDR only)	☾
VS,XXXXX	Assign VMA size of (XXXXX+1) pages to the VMA program.‡	
WS,YYYY	Assign working set size of YYYY pages to the VMA program.‡	
SH,label	EMA area of program is to reside in the specified shareable EMA partition.‡	☾
SA,XX	Reserve XX words of local SAVE area for FORTRAN.‡	

‡FOOTNOTE: Specification of the ‡ commands must precede specification of any RELOCATE or SEARCH command.

## MLLDR SEGMENTATION COMMANDS

M X.Y	Place the following routines (until the next M or D command) in the specified memory-resident node (default=main).	
D X.Y	Place the following routines (until the next M or D command) in the specified disc-resident node (Y must be at least 1).	☾
NA,name	Load the specified routine with the node being currently loaded. The routine must be found in a user specified library.	
SY,name	Load the specified routine with the node being currently loaded. The routine must be found in the system library.	☾

**CREATE MLLDR COMMAND FILE (SGMTR)**

RU,SGMTR,input[,output[,size[,main  
[,segoptions[,loadoptions]]]]]

- input Program's merged relocatable file.
- output Name of the loader command file to be created (default=user's terminal).
- size Maximum pages allowed in a path (default=28).
- main Main entry point of the program (default=first entry point in the input file).
- segoptions "M" or "D" for memory or disc-resident nodes (default=M), or "A" option will cause an NA or SY command to be generated for every module in the program.
- loadoptions Loader opcodes EM, VM, PF or DB can be specified here.

**VERIFY MLLDR COMMAND FILE (SXREF)**

RU,SXREF,command[,list]

- command MLLDR command file name.
- list Output list file name. If specified a cross reference listing will be provided.

**CREATE AN INDEXED LIBRARY FILE (INDXR)**

RU,INDXR[,namr]

- namr Command file namr or input device lu (default=user's terminal).

**GENERAL PURPOSE HELP (CMD)**

RU,CMD[,key[,lu[,input[,NI]]]]]

- key Used in searching the indexed file for a match (1 - 24 characters in length).
- lu Logical unit where the message associated with the key is to be printed.
- input Indexed sequential file which is to be searched (must be created by GENIX).
- NI Non-interactive mode.

## UTILITIES

### CREATE INDEXED SEQUENTIAL FILE (GENIX)

RU,GENIX,input,list,output

- input Input text file of keys and text to be indexed.
- list Output list namr (default=log lu).
- output Type 1 indexed file searched by the CMD utility.

### SOURCE FILE COMPARISON (SCOM)

RU,SCOM,input 1,input 2[,output[,option  
[,match[,char]]]]

- input 1, input 2 Input files to be compared.
- output Output list namr (default=user's terminal).
- option Any of the following, concatenated without intervening commas:
  - F1 List only lines unique to input 1.
  - F2 List only lines unique to input 2.
  - BO List lines common to both files.
  - NN Suppress line numbering.
  - TB Include trailing blanks in match.
  - Dx Use x as a don't-care character.
  - Cy Ignore blank lines with y in column 1.Default is F1F2
- match Number of consecutive lines which must match in the input files before a mismatch is ended (default=3).
- char Maximum record size in the input files (default=156 characters).

### FILE BACK-UP UTILITY (FC)

RU,FC[,command string]

- command string A string of FC commands. If defaulted, commands are entered interactively from the user's terminal.
  - FC commands
    - LL — Set list device.
    - ECHO OFF — Suppress command echoing at list device.

SCRATCH	— Specify disc cartridge to be used for internal scratch files.
TITLE	— Establish title for tapes.
CF	— Establish title for comment file.
CO	— Initiate copy operation.
GR	— Execute the following CO commands as a single operation.
AG	— Halt the GR command.
EG	— End of the group copy commands.
DE	— Set a common group or destination for all the grouped CO commands.
TR	— Transfer to/from the FC command file.
CL(AL)	— List the cartridge list.
DL	— List the directory list.
LC	— List the comment file.
LH	— List the header file from tape.
EX	— Exit FC.
*comment	— Enter comment in command file.
?	— Display a summary of available commands.

### SAVE DISC CARTRIDGE (WRITT)

RU,WRITT ,—lu(c) ,lu(m) ,IH [,DC[,VE[,"..."]  
 ,+crn

—lu(c)	is the logical unit (lu) number of the cartridge to be saved on mag tape.
+crn	is the cartridge reference number (CRN) of the cartridge to be saved on mag tape.
lu(m)	is the logical unit (lu) number of the mag tape unit (default is LU8). Either a positive or negative lu can be specified.
IH	inhibits tape rewind (default is to rewind).
DC	disable overlay check.
VE	verify data transfer.
"..."	comment to be appended to tape header; 40 characters maximum.

## UTILITIES

### RESTORE DISC CARTRIDGE (READT)

RU,READT , -lu(c) ,lu(m) ,P ,size[,IH[,VE or CO]  
,+crn ,G

- lu(c) is the logical unit (lu) number of the cartridge to which the previously saved cartridge is to be restored.
- +crn is the cartridge reference number (CRN) of the cartridge being restored.
- lu(m) is the logical unit (lu) number of the mag tape unit (default is LUB). Either a positive or negative lu can be specified.
- P designates that the cartridge is to be restored as a private cartridge.
- G designates that the cartridge is to be restored as a group cartridge.
- size is the desired size of the cartridge to which the mag tape contents is to be restored. The size is specified in number of tracks (default is the size of the cartridge saved on the mag tape).
- IH inhibits tape rewind (default is to rewind).
- VE verify data transfer.  
Either VE or CO may be included, but not both.
- CO perform word-by-word comparison of tape to restored cartridge.

### FORTRAN AND MACROASSEMBLER

#### MACRO

RU, ,source[,list[,relocatable[,lc[,cs[,work]]]]]  
FTN7X

- source Disc file or LU for source file.
- list Disc file,lu,or "-" for list. "-" creates file 'source for listing if source begins with a & (default=user's terminal).
- relocatable Name of file or "-" for relocatable code. "-" creates file %source for relocatable code if source begins with & (no default).

- lc Line count per page.
- work File name to be used by the Macroassembler for scratch space.
- cs Optional control statement which overrides the source file control statement. Options are as follows:
- FORTTRAN**
- L Output source to list file.
  - A Output Macroassembly listing to list file.
  - T Output symbol table for each main or subprogram to list file.
  - M Output a mixed listing of both the source and the object program to the list file.
  - C Output a cross reference symbol table listing to the list file.
  - F Perform page eject.
  - D Compile debug lines.
  - n Error routine n supplied. n is a decimal digit 1-9 which specifies an error routine ERRn.
  - Q Include the approximate relocatable address of each statement on the listing.
  - I Integers are stored in one word (default).
  - J Integers are stored in two words.
  - X Double precision is stored in three words (default).
  - Y Double precision is stored in four words.
  - S Generate symbolic debug records.
- MACROASSEMBLER**
- A Absolute assembly, the addresses generated by the Macroassembler are interpreted as absolute locations in memory.
  - R Relocatable assembly, the object program may be loaded anywhere in memory.
  - L Output source listing to list file. This includes both the opcode and the address of the operand if it is a memory reference instruction.
  - Q Output source listing to list file. This includes only the operand address for single word memory reference instructions, otherwise the entire object code will be listed.
  - T Output symbol table to list file.
  - C Output a cross reference symbol table to the list file.

## UTILITIES

- I Generate microcode instructions when possible.
- O Generate old records (compatible with RTE-IVB and earlier operating systems).
- N,Z Selective assembly, sections of the program are to be included or excluded at assembly time depending upon the option specified.
- F The floating point machine instructions are to be used instead of the software simulation routines for: FIX,FLT,FDV,FMP,FAD,FSB.
- X No EAU hardware on machine.
- S Generate symbolic debug records.

## PASCAL

**RU,PASCL[,source[,list[,relocatable[,option]]]]**

- source** Disc file for source file (default=user's terminal).
- list** Disc file, lu, or "-" for list. "-" creates file 'source for listing if source begins with & (default=user's terminal).
- relocatable** Name of file or "-" for relocatable code. "-" creates file %source for relocatable code if source begins with & (no default).
- option** Option file containing the number of pages of EMA used by the compiler (default=15). If option=1, enter option at user's terminal.

## COMPL AND CLOAD

**COMPL**  
**RU, ,source[,list[,relocatable[,cs]]]**  
**CLOAD** **or**  
**[,option]]]**

These utilities automatically invoke the appropriate compiler or assembler for a specified source file. CLOAD, in addition, schedules LOADR.

- source** Name of source file.
- list** Disc file, lu, or "-" for list file. "-" creates file 'source for list file if source begins with &. For CLOAD list must be an lu (default=user's terminal).

<b>relocatable</b>	Name of file or "-" for relocatable code. "-" creates file %source for relocatable code if source begins with & (no default).
<b>cs</b>	Optional control statement which overrides the source file control statement.
<b>option</b>	Option file containing a control string option list (Pascal only).

## DBUGR AND MLSDB

### DBUGR/MLSDB COMMAND CONVENTIONS

<b>\</b>	Escape (character mode) or Backslash (block mode).
<b>/</b>	Forward slash.
<b>[ ]</b>	Input control character.
<b>LF</b>	Line feed.
<b>CR</b>	Carriage return.

### BREAKPOINT AND PROGRAM CONTROL

<b>n\B</b>	Set a breakpoint at location n.
<b>\B</b>	List breakpoint table and enter remove breakpoint mode.
<b>\ \B</b>	Remove all breakpoints.
<b>&lt;seg&gt;\B</b>	Set a breakpoint at entry to segment (DBUGR only).
<b>&lt;path&gt;\B</b>	Set a breakpoint at entry to path (MLSDB only).
<b>n&lt;seg&gt;\B</b>	Set a breakpoint in segment at location n (DBUGR only).
<b>n&lt;path&gt;\B</b>	Set a breakpoint in the specified path at location n (MLSDB only).
<b>["A]\B</b>	Break at entry to all segments or paths.
<b>["N]\B</b>	Do not break at entry to all segments or paths.
<b>\P</b>	Proceed with program execution after a break trap.
<b>\ \P</b>	Proceed with conditional breakpoint invoked.
<b>n\P</b>	Proceed; do not trap until n breakpoints from now.
<b>n \ \P</b>	Proceed; do not trap until n breakpoints, including conditional breakpoints.



## UTILITIES

n\G	Continue execution at location n.	
n\X	Execute the instruction n, then return control to DBUGR or MLSDB.	☾
\T	Trace one instruction.	
n\T	Trace n instructions.	
\ \T	Trace an entire subroutine call with no argument list or alternate returns.	☾

## MEMORY EXAMINATION AND MODIFICATION

n<S	Define the symbol S as the value n.	
n/	Print and open location n.	
[LF]	Print and open the next location.	☾
[CR]	Close current location.	
/	Open and print the contents of the location printed to by the last quantity typed.	
\ /	Open and print the contents of the location pointed to by the last quantity typed, only looking at bits 0-10.	
[TAB] or [CNTL I]	Open, set the location counter to, and print the contents of the location pointed to by the quantity typed.	
\ [TAB] or \ [CNTL I]	Open, set the location counter to, and print the contents of the location pointed to by the quantity typed, looking only at bits 0-10.	
m[CR]	Change the contents of location n to m.	☾
m[LF]	Same as m[CR] above and also print and open the next location.	

## SPECIAL REGISTERS

\M	Display contents of the special registers.	
AREG/	Examine and modify A-Register.	☾
BREG/	Examine and modify B-Register.	
XREG/	Examine and modify X-Register.	
YREG/	Examine and modify Y-Register.	
EOREG/	Examine and modify EO-Register.	☾
MASK/	Examine and modify Search Mask.	

CBVAL/	Examine and modify conditional breakpoint value.
CBMASK/	Examine and modify conditional breakpoint mask.
CBADDR/	Examine and modify conditional breakpoint address.
CBTEST/	Examine and modify conditional breakpoint test.
WRTL/	EXEC control word for DBUGR or MLSD output device.
BRFLG/	Break flag; 0 = check for break, 1 = no check.

### MAP EXAMINATION SPECIAL MODE

\J	Put DBUGR or MLSD into special mode.
UM	Display the user map.
SM	Display the system map.
XL	Set up a cross load from an address in the alternate map.
PA	Displays the port A map.
PB	Displays the port B map.
A	Aborts the special mode with no change.




### PRINT MODE CONTROL

\S	Set print mode to symbolic instruction (default).
\!	Set print mode to symbolic instruction until [CR] is entered.
!	Print the last quantity typed as an instruction.
\C	Set print mode to constant.
\=	Set print mode to constant until [CR] is entered.
=	Print the last quantity typed as a constant.
\H	Set print mode to ASCII characters.
\'	Set print mode to ASCII characters until [CR] is entered.
'	Print the last quantity typed as two ASCII characters.
\A	Set print mode to address.
\	Set print mode to address until [CR] is entered.
←	Print the last quantity typed as an address.
n\R	Change the output radix to n.

## UTILITIES

### SYSTEM CONFIGURATION DISPLAY (LUPRN)

RU,LUPRN[,lu[,AL[,SC[,TY[,DV[,??]]]]]]

lu	Logical unit number of the list device (default is user terminal).	
AL	List all devices, sort by system lu.	
LU[:n:n]	List all devices within specified range (optional), sort by system lu.	
SC[:n:n]	List all devices with assigned select codes within specified range (optional), sort by select code.	
TY[:nB:nB]	List all devices within specified octal range (optional), sort by device type.	
DV	List system driver table, followed by AL option listing.	
??	List descriptive summary of LUPRN and runstring options.	

### FOWN

RU,FOWN[mask]

Displays owners of and disc space used by files specified.

mask File mask. Default = all CI files.

### FPACK

RU FPACK lu

Rearrange files on file system volume, increasing largest free space on volume.

lu LU of volume to be packed.

### FREES

RU FREES [lu]

Report total free space and size of largest free space on CI file system volume.

lu LU of CI file system volume. Default = all CI volumes.

**FSCON**

RU FSCON lu

Convert FMGR cartridge to CI (hierarchical) structure.

lu LU of FMGR cartridge to be converted.

**FVERI**

RU FVERI [lu]

Verify that data within a hierarchical file system volume is consistent.

lu LU of volume to be verified. Default = all volumes.

**LINDX**

RU,LINDX,inputFile,outputFile

Index library files for faster searching.

inputFile File to be indexed.

outputFile Indexed library (different from inputFile).

**MERGE**

RU,MERGE,mergeList,file|lu

Combine two or more input files into a single output file.

mergeList File containing names of files to be merged.

file|lu Output file name or LU.

**OLDRE**

RU,OLDRE,namr

Change extended relocatable record formats to nonextendable formats.

namr Name of type 5 file. Begins with %.

## UTILITIES

### TF

#### RU TF [command]

Back up and restore files from the file addressing space to magnetic or CS80 cartridge tape.

<b>command</b>	Default = interactive mode.
<b>?</b>	List commands and their syntax.
<b>COPY</b>	Copy files as specified by parameters.
<b>TITLE</b>	Set title for tape header file.
<b>DEFAULT</b>	Set default source, destination, and options for subsequent COPY commands.
<b>GROUP, EG, AG</b>	Group multiple COPY commands into a single COPY operation.
<b>LL</b>	Set list device or file for LH or DL.
<b>LH</b>	List header file from DF tape.
<b>DL</b>	Compile directory list of TF tape.
<b>TR</b>	Transfer to or return from TF command file.
<b>EX</b>	Exit TF.
<b>*</b>	Comment.

**FLAG**







[RU,]FLAG,pfile[-options],sfile[...]

- pfile** Patterns file; SEP.6 or custom patterns file.
- sfile** Source file or files to be searched.
- options** any combination of the following:
- C** Count: for each word matched, print a count of lines that contain a match.
  - K** Make case significant in determining a match. By default, case is not significant (a matches a or A).
  - M** Print source file name before each output line (default when more than one source file is specified.).
  - N** Print the line number of each matched line before each output line.
  - V** Verbose: print all lines in the file with line numbers. Flag matches in lines and print count for each word.
  - P** Pascal
  - B** BASIC
  - F** FORTRAN
  - A** Assembler
- Language options assume that all files specified after a language option are source code in that language, all comments in the source are ignored.
- Ofile** Specifies output file (default = terminal). The file name must follow the option letter with no space.

## UTILITIES

### EXT

[RU,]EXT[,-options],srcfile[,outfile]

<b>srcfile</b>	Source relocatable (type 5) file to be searched for external references.	
<b>outfile</b>	Output file to receive list of externals found in srcfile. If outfile exists, EXT output is appended to it, unless R option is specified.	
<b>-options</b>	are any of the following:	
<b>C</b>	Condense the output list: separate the listed externals with spaces, rather than <crLf>. Available only if outfile = terminal.	
<b>Lnn</b>	Lengthen output line to nn characters. Default = minimum = 80; maximum = 134. Forces C option; available only if outfile = terminal.	
<b>Snn</b>	Scroll the output nn lines at a time (default = 22), prompt for more after each nn lines. Use -S0<terminalLU> for continuous printing without prompting. Available only if outfile = terminal.	
<b>N</b>	Name: include the nam record of the routine in which the externals are found.	
<b>T</b>	Identify entry point and external names.	
<b>V</b>	Verbose: combine C, N, and T options.	
<b>R</b>	Replace content of outfile if it exists, rather than appending to it.	
<b>Efile</b>	Error message file (default = terminal). If file already exists, messages are appended.	
<b>Ffile</b>	Find only the externals in the named patterns file.	
<b>Ifile</b>	Ignore the externals in file.	

**FPORT**

[RU,]FPORT,-E,tmap[,tdev]

or

[RU,]FPORT,-I[FM],tmap[,tdev]

**-E** specifies Export mode.

**-I** specifies Import mode.

**tmap** is the transport map file defining the files to be exported/imported.

**tdev** is the device LU of the external medium through which tmap and the transported files are copied. The default is to LU 8.

Mutually exclusive flags, applicable in Import mode only:

**F** = Force the use of tmap as the transport map and ignore the transport map in the transport file.

**M** = Map only. Import only the transport map from the transport file into tmap.

Transport Map Flags:

**-F** Treat the export name as an FMGR name.

**-f** Treat the import name as an FMGR name.

**-b** Treat the file as a binary file. This flag has meaning only for HP-UX files.





# EXEC CALLS

CONTENT	PAGE
I/O, READ/WRITE .....	H-3
I/O, CLASS GET .....	H-4
I/O CONTROL .....	H-5
PROGRAM COMPLETION .....	H-7
PROGRAM SUSPEND .....	H-7
PROGRAM SWAP CONTROL .....	H-8
PROGRAM SCHEDULE .....	H-8
STRING PASSAGE .....	H-9
STATUS DEVICE .....	H-9
STATUS PARTITION .....	H-10
MEMORY SIZE .....	H-10
TIME REQUEST .....	H-11
TIMED EXECUTION (ABSOLUTE) .....	H-11
TIMED EXECUTION (OFFSET) .....	H-12
TRACK ALLOCATION .....	H-12
TRACK RELEASE .....	H-13
CLASS OWNERSHIP MANAGEMENT .....	H-13
LU LOCK .....	H-14
RESOURCE MANAGEMENT .....	H-15

EXEC  
CODE

PAGE

1 .....	H-3
2 .....	H-3
3 .....	H-5
4 .....	H-12
5 .....	H-13
6 .....	H-7
7 .....	H-7
8 .....	H-8
9 .....	H-8
10 .....	H-8
11 .....	H-11
12 .....	H-11
13 .....	H-9
14 .....	H-9
15 .....	H-13
16 .....	H-13
17 .....	H-3
18 .....	H-3
19 .....	H-5
20 .....	H-3
21 .....	H-4
22 .....	H-8
23 .....	H-8
24 .....	H-8
25 .....	H-10
26 .....	H-10



**I/O,READ/WRITE**EXEC  
1,2,17,18,20**CALL EXEC (ICODE,ICNWD,IBFR,ILEN  
[IPRM1],[IPRM2],ICLAS)**

<b>ICODE</b>	1 = READ 2 = WRITE 17 = Class READ 18 = Class WRITE 20 = Class WRITE/READ
<b>ICNWD</b>	Control word, see I/O Control for format. If Z bit (12) is set, an additional control buffer specified by IPRM1,IPRM2 is passed to the driver or to the program doing the GET call.
<b>IBFR</b>	Data buffer.
<b>ILEN</b>	Data length (+ words, - chars).
<b>IPRM1</b>	Optional, or disc track number (for disc transfers), address of additional control buffer (if Z bit is set), or high word of block (for CS80 discs).
<b>IPRM2</b>	Optional, or disc sector (for disc transfers), length of additional control buffer (if Z bit is set), or low word of block (for CS80 discs).
<b>ICLAS</b>	Class number — required with Class I/O only. Class number is allocated and assigned an owner by a call to the CLRQ subroutine. The class number can also be allocated by the EXEC 17,18,20 call with ICLAS=0.

**Returns**

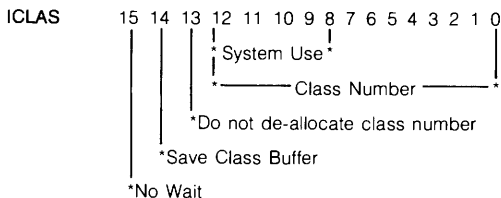
<b>Normal I/O</b>	A = Status, EQT wd. 5 (if unbuffered device). B = Transmission log (if unbuffered device).
<b>Class I/O</b>	A = 0 — Request completed. A = -1 — No class number (if no wait bit is set). A = -2 — No memory or buffer limit exceeded (if no wait bit is set). B = Meaningless.

**I/O, CLASS GET**

EXEC

21

CALL EXEC (21, ICLAS, IBUFR, ILEN[, IP1][, IP2][, IP3])



IBUFR Data buffer.

ILEN Buffer length (+ words, - characters).

IP1 IPRM1 value returned from a class READ/WRITE or CONTROL call.

IP2 IPRM2 value returned from a class READ/WRITE or CONTROL call.

IP3 Returned value of original request code (ICODE).

1 = 17/20 (READ, WRITE/READ)

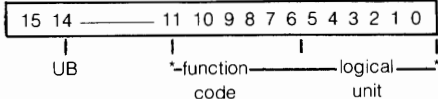
2 = 18 (WRITE)

3 = 19 (CONTROL)

**Returns****A-register** If data, then A15 = 0 and A = status (EQT wd. 5).

If no data, and no wait bit is set, then A15 = 1 and A = -(numb + 1) where numb is number of requests made to class but not yet serviced by driver.

**B-register** If data, then B = transmission log (positive words or characters depending on original request). If no data, then B = meaningless.





**I/O CONTROL**EXEC  
3,19CALL EXEC(ICODE, ICNWD<, IPRAM>  
, ICLAS[, IOP1][, IOP2])ICODE      3 = Control  
            19 = Class ControlICNWD      Control word, see Function Codes below for octal  
            bits 6-10.

UB            Unbuffered bit.

IPRAM        Optional or required for some control functions.

TTY  
n space n lines  
0 no line feedLINE PRINTER  
+n space n lines  
-n top-of-form  
0 no line feedICLAS        Class number — required with class control only.  
Class number is allocated and assigned an owner  
by a call to the CLRQ subroutine. The class number  
can also be allocated by the EXEC 19 call with  
ICLAS=0.IOP1        (when ICODE = 19) Passed through to Class I/O  
IOP2        GET request.**Returns**Normal I/O    A = Status, EQT wd. 5 (if unbuffered device).  
                  B = Meaningless

## EXEC CALLS

Class I/O	A = Class number B = Meaningless	
Function Code	ICNWD Octal-bits 6-10. See particular driver manual for more information.	
	00 Clear device	
	01 Write end-of-file (MT,CTU)	
	02 Backspace one record (MT,CTU)	
	03 Forward space one record (MT,CTU)	
	04 Rewind (MT,CTU)	
	05 Rewind standby (MT,REWIND CTU)	
	06 Actual status of device (MT,CTU)	
	07 Set end-of-paper tape	
	10 Generate paper tape leader.	
	11 List output line spacing, use IPRAM	
	12 Write gap in case of error (MT)	
	13 Forward space one file (MT,CTU)	
	14 Backward space one file (MT,CTU)	
	15 Conditional top-of-form (LP)	
	20 Enable terminal (CRT)	
	21 Disable terminal (CRT)	
	22 Set time-out, use IPRAM (CRT)	
	23 Ignore further requests until:	
	a) Device queue empty	
	b) Input request encountered	
	c) Restore Control request received	
	24 Restore output processing	
	26 Write end-of-data (CTU)	
	27 Locate file number, use IPRAM (CTU)	
	30 Block addressing cached access (CS80) discs	
	34 Block addressing (CS80) discs	

**PROGRAM COMPLETION**EXEC  
6

CALL EXEC (6 [,INAME][,INUMB][,IPRM1,...,IPRM5])

CALL RMPAR(IPRM1,...IPRM5) parameter pick-up.

<b>INAME</b>	Terminate INAME or if 0, terminate calling program.
<b>INUMB</b>	0 Normal completion (default). -1 Serial reusability. 1 <b>Terminate saving resources.</b> 2 Terminate on next schedule: save tracks. 3 Terminate immediately and release tracks.

**IPRM1-IPRM5** Up to 5 optional parameters passed to caller next time he executes (INAME = 0 only).

**Returns**

<b>A-register</b>	Unchanged.
<b>B-register</b>	Unchanged or address of optional parameters (if specified).

**PROGRAM SUSPEND**EXEC  
7

CALL EXEC (7)

If program is rescheduled with a GO command that includes parameters, use RMPAR for parameter pick up.

<b>A-register</b>	Unchanged.
<b>B-register</b>	Unchanged or parameter address.



## PROGRAM SWAP CONTROL

EXEC  
22

## CALL EXEC (22,IOPTN)

IOPTN	0 Swap;
	1 Do not swap.

## Returns

A-register	Meaningless
B-register	Unchanged

## PROGRAM SCHEDULE

EXEC  
8,9,10,23,24CALL EXEC (ICODE,INAME[,IPRM1,  
...,IPRM5][,IBUFR,ILEN])

ICODE	8 = Segment load
	9 = Immediate, wait
	10 = Immediate, no wait
	23 = Queue, wait
	24 = Queue, no wait

INAME	Name of program or segment to be scheduled.
-------	---

IPRM1- IPRM5	Up to 5 optional parameters passed to program specified in INAME.
-----------------	---

IBUFR	Buffer to pass to son. Not used for EXEC 8.
-------	---

ILEN	Length of buffer (+ words, - characters). Son recovers buffer using String Passage (ICODE = 14) EXEC call. Not used for EXEC 8.
------	---

## Returns

A-register	0 if schedule successful.
	Program status if son not scheduled (immediate schedule only).
	If EXEC 8, the segment's ID segment address.

B-register	Unchanged, or address of IPRM1-IPRM5 if they were used.
------------	---

**STRING PASSAGE**EXEC  
14**CALL EXEC (14,IRCOD,IBUFR,ILEN)**

IRCOD	Retrieve/write code: 1 Retrieve buffer or command string. 2 Write buffer to father.
IBUFR	Buffer location.
ILEN	Buffer length (+ words, - characters).

**Returns**

A-register	0 = successful; 1 = no string found.
B-register	Transmission log.

**STATUS, DEVICE**EXEC  
13**CALL EXEC (13,ICNWD,IST1[,IST2][,IST3])**

ICNWD	Lu of device.
IST1	Returned value of EQT word 5, see Device Status table.
IST2	Returned value of EQT word 4, see EQT table.
IST3	Returned value specifying whether device is "up" or "down".

**Returns**      Meaningless.

**STATUS, PARTITION**EXEC  
25CALL EXEC (25, IPART, IPAGE, IPNUM, ISTAT)

- IPART** Partition number.
- IPAGE** Returned value of starting page number.
- IPNUM** Returned value of the number of pages with base page included (-1 returned if illegal partition number).
- ISTAT** Return for partition status:

15	14	13	12	11	10	—	7	—	0
----	----	----	----	----	----	---	---	---	---

RS RT M S C E-0-ID SEG NO.

- RS = 1 if partition reserved
- RT = 1 if partition is real time
- M = 1 if partition is mother
- S = 1 if partition is subpartition
- C = 1 if chain is in effect
- E = 1 if partition is shareable EMA partition

## Returns

- A-register** Meaningless.
- B-register** Unchanged.

**MEMORY SIZE**EXEC  
26CALL EXEC (26, IFAW, ILMEM, INPGS[, IMAP])

- IFAW** Returned value of first available word address after program.
- ILMEM** Returned value, the number of words between end of program and end of program's address space.
- MPGS** Returned value, number of pages in partition.
- IMAP** Returned value of user map (32 word array).

## Returns

- A-register** Meaningless.
- B-register** Unchanged.

**TIME REQUEST**EXEC  
11**CALL EXEC (11, ITIME [, IYEAR])****ITIME** Return for time value as follows:

ITIME (1) = 10's of milliseconds

ITIME (2) = Seconds

ITIME (3) = Minutes

ITIME (4) = Hours

ITIME (5) = Julian day of year

**IYEAR** Returned value of year (e.g., 1975) (optional).**Returns****A-register** Meaningless.**B-register** Unchanged.**TIMED EXECUTION**

(Absolute Start)

EXEC  
12**CALL EXEC (12, INAME, IRESL, IMULT,  
IHRS, IMIN, ISEC, IMSEC)****INAME** Schedule INAME or if 0, schedule calling program.**IRESL** Resolution code, see initial offset EXEC 12.**IMULT** Execution multiple (set= 0 means run once).**IHRS****IMIN****ISEC****IMSEC**

} Defines absolute start time.

**Returns****A-register** Meaningless.**B-register** Unchanged.

## EXEC CALLS

### TIMED EXECUTION

EXEC

(Initial Offset)

12

#### CALL EXEC (12, INAME, IRESL, IMULT, IOFST)

INAME Schedule INAME or if 0, schedule calling program.

IRESL Resolution code.

1 = 10's/ms

2 = Seconds

3 = Minutes

4 = Hours

IMULT Execution multiple (set = 0 means run once).

IOFST Relative start time (negative value) from current time.

#### Returns

A-register. Meaningless.

B-register Unchanged.

### TRACK ALLOCATION

EXEC

4,15

#### CALL EXEC (ICODE, ITRAK, ISTRK, IDISC, ISECT)

ICODE 4 = local.  
15 = global.

ITRAK Number of tracks.

B15 = 1 — Program not suspended if tracks not available.

B15 = 0 — Program suspended if tracks not available.

ISTRK Returned value of starting track number (-1 if tracks not available.)

IDISC Returned value of disc lu, where tracks were allocated.

ISECT Returned value of number of sectors per track.

Returns Meaningless.

**TRACK RELEASE**EXEC  
5,16**CALL EXEC (ICODE,ITRAK[,ISTRK][,IDISC])**

**ICODE**        5 = local.  
                  16 = global.

**ITRAK**        Number of tracks (If ICODE=5, then -1 = all tracks, ISTRK and IDISC unnecessary.)

**ISTRK**        Starting track number.

**IDISC**        Disc lu.

**Returns**      Local.

**A-register**    Meaningless.

**B-Register**    Meaningless.

**Returns**      Global

**A-register**    Status.

0 = Tracks released.

-1 = No tracks released, one in use.

-2 = No tracks released, one not global.

**B-register**    Meaningless.

**CLASS OWNERSHIP MANAGEMENT****CALL CLRQ (IFUNC,ICLAS[,IOP1])**

**IFUNC** = Class management control function.

1 = Class ownership assigned.

2 = Flush class requests deallocating the class.

3 = Flush class requests on lu specified in IOP1.

Bit 14 = No-abort bit

Bit 15 = No-wait bit

**ICLAS** = Class number

**IOP1** = Call dependent parameter; used to describe a program name or lu.

**LOGICAL UNIT LOCK PROGRAM CALL****CALL LURQ (IOPTN,LUARY,NOLU)**

- IOPTN** Octal control word as follows:
- 0x0000 = Unlock specified lu's.
  - 1x0000 = Unlock all lu's program currently has locked.
  - 0x0001 = Lock with wait specified lu's.
  - 1x0001 = Lock without wait specified lu's.
- x(bit 14) is no abort bit; 1 = don't abort.
- LUARY** Array of lu's to be locked/unlocked. Ignored when IOPTN = 1x0000.
- NOLU** Number of lu's to be locked/unlocked. Ignored when IOPTN = 1x0000.

**Returns**

- A-register** 0 = Lock successful.  
-1 = RN not available.  
1 = lu already locked.
- B-register** Unchanged.

<b>RESOURCE MANAGEMENT</b>
----------------------------

**CALL RNRQ (ICODE,IRN,ISTAT)**

<b>ICODE</b>	Control word as follows: Bits 15 no wait. 14 no Abort. 13 } . } reserved for system use. . } . } . } . } 5 clear } 4 global } allocate option. 3 local } 2 clear } 1 global } set option. 0 local }
<b>IRN</b>	Resource number.
<b>ISTAT</b>	Status word. 0 = Normal deallocate return. 1 = RN is clear (unlocked). 2 = RN is locked locally to caller. 3 = RN is locked globally. 4 = No RN available now. 6 = RN locked locally to other program. 7 = RN was locked globally when request was made.

**Returns**

<b>A-register</b>	Meaningless.
<b>B-register</b>	Unchanged.





# CI FILE HANDLING



CONTENT	PAGE
PARAMETERS USED IN THIS SECTION .....	I-4
STANDARD FMP SUBROUTINES .....	I-4
FmpClose .....	I-4
FmpOpen .....	I-4
FmpPurge .....	I-5
FmpRead .....	I-5
FmpOpenScratch .....	I-6
FmpPosition .....	I-6
FmpPost .....	I-6
FmpRecordCount .....	I-7
FmpRecordLen .....	I-7
FmpRename .....	I-7
FmpRewind .....	I-7
FmpSetEof .....	I-7
FmpSetPosition .....	I-8
FmpSize .....	I-8
FmpTruncate .....	I-8
FmpWrite .....	I-8
HIERARCHICAL FILE SYSTEM FMP SUBROUTINES .....	I-9
Calc_Dest_Name .....	I-9
FattenMask .....	I-9
FmpAccessTime .....	I-9
FmpBuildHierarch .....	I-10
FmpBuildName .....	I-10
FmpBuildPath .....	I-10
FmpCreateDir .....	I-11
FmpCreateTime .....	I-11
FmpEndMask .....	I-11

CONTENT	PAGE
FmpHierarchName .....	I-11
FmpInfo .....	I-12
FmpInitMask .....	I-12
FmpLastFileName .....	I-12
FmpMaskName .....	I-12
FmpNextMask .....	I-13
FmpOpenFiles .....	I-13
FmpOwner .....	I-14
FmpParseName .....	I-14
FmpParsePath .....	I-14
FmpProtection .....	I-14
FmpSetDirInfo .....	I-15
FmpSetOwner .....	I-15
FmpSetProtection .....	I-15
FmpSetWorkingDir .....	I-16
FmpShortName .....	I-16
FmpStandardName .....	I-16
FmpUnPurge .....	I-16
FmpUpdateTime .....	I-16
FmpWorkingDir .....	I-16
MaskOldFile .....	I-17
MaskOpenId .....	I-17
MaskMatchLevel .....	I-17
MaskSecurity .....	I-17
WildCardMask .....	I-18
UTILITY FMP SUBROUTINES .....	I-18
DcbOpen .....	I-18
FmpAppend .....	I-18
FmpControl .....	I-18
FmpCopy .....	I-18
FmpDevice .....	I-19
FmpDismount .....	I-19
FmpEof .....	I-19
FmpError .....	I-20
FmpExpandSize .....	I-20
FmpFileName .....	I-20
FmpInteractive .....	I-20
FmploOptions .....	I-20
FmploStatus .....	I-21
FmpList .....	I-21
FmpMount .....	I-22
FmpPackSize .....	I-22
FmpReadString .....	I-22
FmpReportError .....	I-23

CONTENT	PAGE
FmpRpProgram .....	I-23
FmpRunProgram .....	I-23
FmpRwBits .....	I-24
FmpSetDcbInfo .....	I-24
FmpSetIoOptions .....	I-24
FmpSetWord .....	I-24
FmpUniqueName .....	I-25
FmpWriteString .....	I-25
<b>SPECIAL-PURPOSE DS COMMUNICATION</b>	
SUBROUTINES .....	I-25
DsCloseCon .....	I-25
DsDcbWord .....	I-26
DsDiscInfo .....	I-26
DsDiscRead .....	I-26
DsFstat .....	I-27
DsOpenCon .....	I-27
DsSetDcbWord .....	I-27

### PARAMETERS USED IN THIS SECTION

The following parameters are used throughout this section. They are not described beneath the calls that use them unless additional information must be given for the call.

- |              |   |   |
|--------------|---|---|
| <b>error</b> | Returns negative code, or 0 if no error occurs.                             |  |
| <b>dcb</b>   | Integer array (at least 16 words) containing file data control block (DCB). |  |

### STANDARD FMP SUBROUTINES

#### FmpClose






**Purpose:** Close file (make it inaccessible).

**Syntax:** error = FmpClose(dcb,error)

#### FmpOpen

**Purpose:** Open file for access. Nonexistent file is created if fileDescriptor is complete.

**Syntax:** type = FmpOpen(dcb,error,fileDescriptor,options, buffers)

- |                |  |  |
|----------------|--|--|
| <b>type</b>    | Nonnegative integer. Returns file type or error code.  |  |
| <b>options</b> | Character string. Selects file open options from the following:<br><b>Access mode:</b><br>R: open for reading<br>W: open for writing<br><b>File Existence:</b><br>O: open an existing file<br>C: create a new file<br><b>Miscellaneous:</b><br>S: open a shared file<br>U: open in update mode<br>T: file is temporary<br>F: force type to 1 for unbuffered access<br>X: access extents in type 1 or 2 file<br>D: fileDescriptor specifies a directory | <br><br> |

Q: open file quickly, do not record access time

Options can be in any order, upper or lowercase.

**buffers** DCB buffer size, in range 1 to 127.

## FmpPurge

**Purpose:** Purge a file.

**Syntax:** error = FmpPurge(fileDescriptor)

## FmpRead

**Purpose:** Read from a file.

**Syntax:** length = FmpRead(dcb,error,buffer,maxlength)

**length** Returns number of bytes actually read, or negative error code. If more than 32767 bytes are read, length may be negative although no error occurred.

**buffer** Word-aligned buffer into which data file is to be transferred.

**maxlength** Maximum number of bytes to read. Treated as unsigned single integer from 0 to 65534.

## CI FILE HANDLING

### FmpOpenScratch

**Purpose:** An interface to FmpOpen; standardizes scratch file creation.

**Syntax:** type = FmpOpenScratch(dcb,error,filedescriptor,  
options,buffers,nameused)

**type** See FmpOpen.

**options** Same as FmpOpen, plus the following:  
Z: pass filedescriptor to FmpUniqueName to  
create a unique scratch file descriptor.

**nameused** Character string. Returns the file descriptor  
used in call to FmpOpen.

**buffers** Same as FmpOpen.

### FmpPosition

**Purpose:** Return current file position.

**Syntax:** error = FmpPosition(dcb,error,record,position)

**record** Returns current record number.

**position** Returns current internal file position.

### FmpPost

**Purpose:** Post data to a file.

**Syntax:** error = FmpPost(dcb,error)

## FmpRecordCount

**Purpose:** Return number of records in file.

**Syntax:** `error = FmpRecordCount(fileDescriptor,nrecords)`

**nrecords** Returns number of records in file. For file types 1 and 2, this is the maximum number of records that the file can accommodate. For file types 3 and above, this is the number of records before EOF (may be inaccurate if file is open for writing).

## FmpRecordLen

**Purpose:** Return length of longest record in file.

**Syntax:** `error = FmpRecordLen(fileDescriptor,len)`

**len** Length of longest record in file. For file types 3 and above, length of longest record ever written to file even if it has been overwritten.

## FmpRename

**Purpose:** Change file name.

**Syntax:** `error = FmpRename(name1,err1,name2,err2)`

**name1** Name of existing, closed file.

**err1** Error associated with the file name1.

**name2** New name, including security code and directory.

**err2** Error associated with the file name2.

## FmpRewind

**Purpose:** Position file at its first record.

**Syntax:** `error = FmpRewind(dcb,error)`

## FmpSetEof

**Purpose:** Set end-of-file mark at current file position.

**Syntax:** `error = FmpSetEor(dcb,error)`



## CI FILE HANDLING

### FmpSetPosition

**Purpose:** Change file position.

**Syntax:** error = FmpSetPosition(dcb,error,record,position)

**record** Number of record at which file is to be positioned.

**position** If positive, desired internal file position; if negative, desired record number.

### FmpSize

**Purpose:** Return physical file size.

**Syntax:** error = FmpSize(fileDescriptor,size)

**size** Returns physical size.

### FmpTruncate

**Purpose:** Truncate file.

**Syntax:** error = FmpTruncate(dcb,error,blocks)

**blocks** Minimum number of blocks to which file will be truncated.

### FmpWrite

**Purpose:** Write to a file.

**Syntax:** length = FmpWrite(dcb,error,buffer,maxlength)

**length** Number of bytes actually transferred, or a negative error code. If more than 32767 bytes are transferred, length may be negative although no error occurred.

**buffer** Word-aligned buffer containing data to be transferred.

**maxlength** Maximum number of bytes to write; interpreted as unsigned one-word integer from 0 to 65534.

## HIERARCHICAL FILE SYSTEM FMP SUBROUTINES

### Calc\_Dest\_Name

**Purpose:** Create destination file name from a file name, match level, and destination mask.

**Syntax:** Call Calc\_Dest\_Name(sourcename,matchlevel,  
destmask,destname)

**sourcename**

Character string containing full source file-Descriptor.

**matchlevel**

Output of MaskMatchLevel routine.

**destmask** Character string specifying destination mask.

**destname** Character string that returns full destination fileDescriptor.

### FattenMask

**Purpose:** Modify mask.

**Syntax:** Call FattenMask(mask,how)

**mask** Character string specifying mask to be modified.

**how** Integer specifying how to modify mask. If bit 0 is set, "D" is appended to the qualifier; if bit 1 is set and the mask is blank, neither the name nor the type extension will have "@" inserted.

### FmpAccessTime

**Purpose:** Return time of last access for a file.

**Syntax:** error = FmpAccessTime(fileDescriptor,time)

**time** Returns time of last access, expressed as number of seconds since Jan 1, 1970.

## CI FILE HANDLING

### FmpBuildHierarch

**Purpose:** Build file descriptor in hierarchical format.

**Syntax:** Call FmpBuildHierarch(fileDescriptor,dirpath,name,  
typex,qual,sc,type,size,rl,ds)

Parameters are the same as for FmpBuildPath.

### FmpBuildName

**Purpose:** Build file descriptor from given file specifiers.

**Syntax:** Call FmpBuildName(fileDescriptor,name,typex,  
sc,dir,type,size,rl,ds)

#### fileDescriptor

64-character string containing returned file descriptor.

**name** Character string (up to 64 characters) specifying filename.

**typex** Character string (up to 4 characters) specifying type extension.

**sc** Security code (FMGR files).

**dir** Character string (up to 16 characters) specifying directory name.

**type** FMP file type.

**size** File size in blocks.

**rl** Record length.

**ds** Character string (up to 63 characters) specifying DS node name, a user name, or both.

### FmpBuildPath

**Purpose:** Build character string file mask or file descriptor from given file specifiers.

**Syntax:** Call `FmpBuildPath(fileDescriptor,dirpath, name,typex,qual,sc,type, size,r1,ds)`

File specifiers are as described for `FmpBuildName`, except:

**dirpath** Character string (up to 63 characters) naming directory/subdirectory path. Dirpath must end with "/", and must have "/" between each of the directories and subdirectories.

**qual** Character string mask qualifier (up to 40 characters).

### **FmpCreateDir**

**Purpose:** Create directory.

**Syntax:** `error = FmpCreateDir(name,lu)`

**name** Name of directory to be created.

**lu** Disc LU on which to create directory.

### **FmpCreateTime**

**Purpose:** Return time that a file was created.

**Syntax:** `error = FmpCreateTime(fileDescriptor,time)`

**time** Returns time that file was created, expressed in seconds since January 1, 1970.

### **FmpEndMask**

**Purpose:** Close files associated with mask search.

**Syntax:** Call `FmpEndMask(dirdcb)`

**dirdcb** Integer array initialized by `FmpInItMask`.

### **FmpHierarchName**

**Purpose:** Convert file descriptor to hierarchical format (e.g., /dir/subdir/filename).

## CI FILE HANDLING

**Syntax:** Call FmpHierarchName(fileDescriptor)

### FmpInfo

**Purpose:** Return directory information for a file.

**Syntax:** error = FmpInfo(dcb,error,info,flag)

**info** 32-word integer array in which directory information is returned.

**flag** 0 for FMGR file, nonzero for hierarchical file.

### FmpInitMask

**Purpose:** Initialize file structures for FMP mask calls.

**Syntax:** error = FmpInitMask(dirdcb,error,mask,dirOpenName,  
dcbLen)

**dirdcb** Control array, to be used only with FmpNextMask.

**mask** Character string specifying set of files.  
Format is:

dirpath/name.typex,qual:sc:dir:type:size:rl

**dirOpenName**

Returns character string directory path.

**dcbLen** Length of dirdcb, in words.

### FmpLastFileName

**Purpose:** Return last file name in path.

**Syntax:** subroutine FmpLastFileName(fileDescriptor,lastName)

**lastName** Returns filename, a portion of fileDescriptor.

### FmpMaskName

**Purpose:** Build full name for file that matches mask.

**Syntax:** Call FmpMaskName(dirdcb,newname,entry,curpath)

- dirdcb** Control array initialized by FmpInInitMask.
- newname** Character string that returns fileDescriptor.
- entry** 32-word directory entry from FmpNextMask.
- curpath** Character string directory path from FmpNextMask.

## FmpNextMask

**Purpose:** Return directory entry of next file that matches mask.

**Syntax:** `more = FmpNextMask(dirdcb,error,curpath,entry)`

**more** Boolean variable indicating whether search can continue. True if there is another entry to be searched, whether or not an error occurred (if error did occur, current entry is invalid). False if error prevents continuation of search, or when search is complete.

**dirdcb** Control array initialized by FmpInInitMask.

**curpath** Returns character string directory path.

**entry** 32-word array which returns directory entry for each file found.

## FmpOpenFiles

**Purpose:** Indicate open files in a directory.

**Syntax:** `error = FmpOpenFiles(dcb,error,loc,flag)`

**dcb** Directory open for reading.

**loc** Returns directory position of next open file. Caller initializes it 0 to indicate that this is the first call. When all open files in directory are reported, `loc = -1`.

**flag** Returns flag value for a file.

## CI FILE HANDLING

### FmpOwner

**Purpose:** Return name of directory owner.

**Syntax:** error = FmpOwner(dir,owner)

**dir** Directory.

**owner** Character string. Returns log-on name of the user who owns directory.

### FmpParseName

**Purpose:** Separate character string file descriptor into file specifiers.

**Syntax:** Call FmpParseName(fileDescriptor,name,typex,sc,dir,  
type,size,rl,ds)

Parameters are the same as for FmpBuildName.

### FmpParsePath

**Purpose:** Separate character string file mask or file descriptor into file specifiers.

**Syntax:** Call FmpParsePath(fileDescriptor,dirpath,name,typex,  
qual,sc,type,size,rl,ds)

Parameters are the same as for FmpBuildPath.

### FmpProtection

**Purpose:** Return kinds of access available for file or directory (R = read, W = write, RW = both).

**Syntax:** error = FmpProtection(fileDescriptor,ownerAccess,  
othersAccess)

**ownerAccess**

Owner's file access rights.

**othersAccess**

File access rights of users other than owner.

## FmpSetDirInfo

**Purpose:** Change directory information.

**Syntax:** error = FmpSetDirInfo(dcb,error,ctime,atime,utime,  
bbit,prot)

<b>ctime</b>	Create time.
<b>atime</b>	Access time.
<b>utime</b>	Update time.
<b>bbit</b>	Back-up bit.
<b>prot</b>	File protection.

If a parameter is negative, the corresponding value in the directory entry is not changed.

## FmpSetOwner

**Purpose:** Change name of directory owner. Caller must own directory or be a superuser.

**Syntax:** error = FmpSetOwner(dir,err1,owner,err2)

<b>dir</b>	Directory name.
<b>err1</b>	Returns errors associated with dir.
<b>owner</b>	Name of new owner.
<b>err2</b>	Returns errors associated with owner.

## FmpSetProtection

**Purpose:** Change kinds of access available for file or directory (R = read, W = write, RW = both).

**Syntax:** error = FmpSetProtection(fileDescriptor,ownerAccess,  
othersAccess)

**ownerAccess**  
Owner's file access rights.

**othersAccess**  
File access rights of users other than owner.



## CI FILE HANDLING

### FmpSetWorkingDir

**Purpose:** Change working directory.

**Syntax:** error = FmpSetWorkingDir(fileDescriptor)

### FmpShortName

**Purpose:** Return short version of file descriptor, without type, size, or record length.

**Syntax:** error = FmpShortName(dcb,error,fileDescriptor)

### FmpStandardName

**Purpose:** Convert file descriptor to standard format (e.g., subdir/ filename::dir).

**Syntax:** Call FmpStandardName(fileDescriptor)

### FmpUnPurge

**Purpose:** Restore directory information for a file (undo purge).

**Syntax:** error = FmpUnPurge(fileDescriptor)

### FmpUpdateTime

**Purpose:** Return last update time for a file.

**Syntax:** error = FmpUpdateTime(fileDescriptor,time)

**time** Returns time of last update, expressed in seconds since Jan 1, 1970.

### FmpWorkingDir

**Purpose:** Return current working directory.

**Syntax:** error = FmpWorkingDir(name)

**name** Returns name of current working directory.

## MaskOldFile

**Purpose:** Determine if file is a FMGR file.

**Syntax:** `bool = MaskOldFile(dirdcb)`

**bool** Returns true if last file returned by `FmpNextMask` is a FMGR file.

**dirdcb** Integer array initialized by `FmpInitMask`.

## MaskOpenId

**Purpose:** Return D.RTR open flag of last file returned by `FmpNextMask`.

**Syntax:** `openid = MaskOpenId(dirdcb)`

**openid** D.RTR open flag of last file returned by `FmpNextMask`; 0 if file is closed.

**dirdcb** Integer array initialized by `FmpInitMask`.

## MaskMatchLevel

**Purpose:** Return directory level of last file matched.

**Syntax:** `matchlevel = MaskMatchLevel(dirdcb)`

**matchLevel** Directory level in which last file was matched.

**dirdcb** Integer array initialized by `FmpInitMask`.

## MaskSecurity

**Purpose:** Return security code of last FMGR file returned by `FmpNextMask`.

**Syntax:** `seccode = MaskSecurity(dirdcb)`

**seccode** Security code of last file returned by `FmpNextMask` if file was an FMGR file; 0 for FMP file.

**dirdcb** Integer array initialized by `FmpInitMask`.

## CI FILE HANDLING

### WildcardMask

**Purpose:** Check for wildcard characters in mask.

**Syntax:** wild = WildCardMask(mask)

**wild** Returns true if mask refers to more than one file; false otherwise.

**mask** Character string containing mask to be checked.

## UTILITY FMP SUBROUTINES

### DcbOpen

**Purpose:** Indicate whether DCB is open.

**Syntax:** error = DcbOpen(dcb,error)

**error** 0 if dcb is open; negative error code if not.

### FmpAppend

**Purpose:** Position file at EOF mark.

**Syntax:** error = FmpAppend(dcb,error)

### FmpControl

**Purpose:** Issue control request to LU.

**Syntax:** error = FmpControl(dcb,error,param\*4)

**dcb** Must be associated with a device.

**param\*4** Up to 4 parameters (device-dependent).

### FmpCopy

**Purpose:** Copy a file to another file.

**Syntax:** error = FmpCopy(name1,err1,name2,err2,buffer,  
                          bufen,options)

<b>name1</b>	Character string specifying source file. Can be an LU.
<b>err1</b>	Returns errors associated with name1.
<b>name2</b>	Character string specifying destination file. Can be an LU.
<b>err2</b>	Returns errors associated with name2.
<b>buffer</b>	Character buffer (at least 288 words) containing source and destination DCBs and DCB buffers.
<b>buflen</b>	Buffer length, in words.
<b>options</b>	Character string that selects copy options, which are: b = binary a = ASCII p = purge source after copy d = overwrite existing file

### FmpDevice

<b>Purpose:</b>	Indicate whether a DCB is associated with a device file.
<b>Syntax:</b>	bool = FmpDevice(dcb)
<b>bool</b>	True (-1) if dcb is associated with a device file; false (0) if dcb is associated with a disc file or is closed.

### FmpDismount

<b>Purpose:</b>	Dismount a volume.
<b>Syntax:</b>	error = FmpDismount(lu)
<b>lu</b>	LU of volume to be dismounted.

### FmpEof

<b>Purpose:</b>	Return position of EOF mark.
<b>Syntax:</b>	error = FmpEof(fileDescriptor,eofPos)
<b>eofPos</b>	Current internal file position.

## CI FILE HANDLING

### FmpError

**Purpose:** Return error message for FMP error code.

**Syntax:** Call FmpError(error,message)

**message** Character string that returns an error message. If no message is associated with the error identified by the error parameter, a generic error message in the form "FMP error -XXX" is returned.

### FmpExpandSize

**Purpose:** Unpack file size word into double integer.

**Syntax:** blocks = FmpExpandSize(size)

**blocks** Number of blocks in file, in double integer.

**size** File size, in one word.

### FmpFileName

**Purpose:** Return full path name of file.

**Syntax:** error = FmpFileName(dcb,error,fileDescriptor)

### FmpInteractive

**Purpose:** Indicate whether DCB is associated with an interactive device.

**Syntax:** bool = FmpInteractive(dcb)

**bool** True (-1) if dcb is associated with an interactive device; false (0) otherwise.

### FmpIoOptions

**Purpose:** Return I/O option word.

**Syntax:** error = FmploOptions(dcb,error,options)

**options** Character string that selects copy options, which are:

- b = binary
- a = ASCII
- p = purge source after copy
- d = overwrite existing file

## FmploStatus

**Purpose:** Return A and B register values.

**Syntax:** Call FmploStatus(areg,breg)

**areg** Returns value of A-register.

**breg** Returns value of B-register.

## FmpList

**Purpose:** List file to specified LU.

**Syntax:** error = FmpList(fileDescriptor,lu,option,rec1,rec2)

**lu** Output LU.

**option** Character string that selects output format:

- a ASCII
- b binary output displayed as octal

Output format defaults are:

File Type	Format
0, 3, 4	ASCII
1, 2, 5 and up	Binary

**rec1** First record to be listed.

**rec2** Last record to be listed.

If rec1 = 0 and rec2 = 0, whole file is listed.


## CI FILE HANDLING

### FmpMount

**Purpose:** Mount a volume. 


**Syntax:** error = FmpMount(lu,flag,blks)

**lu** LU of disc volume.

**flag** Determines whether to initialize disc before mounting it. The values of flag are: 

1 Do not initialize before mounting.

1 Initialize if disc does not have valid directory.

2 Initialize disc before mounting. 

**blks** Number of blocks to leave free at beginning of volume.

### FmpPackSize

**Purpose:** Pack double integer file size into one word.

**Syntax:** size = FmpPackSize(doublesize)


**size** Returns file size in one word.


**doublesize** File size in double integer.

### FmpReadString

**Purpose:** Read character string from file. 

**Syntax:** length = FmpReadString(dcb,error,string)

**length** Returns positive number of bytes transferred, or negative error code. 

**string** Character string (up to 256 bytes) into which data is transferred. 

## FmpReportError

**Purpose:** Print error message for FMP error code on LU 1.

**Syntax:** Call FmpReportError(error,fileName)

**fileName** Name of file to include with error message.

## FmpRpProgram

**Purpose:** Restore program.

**Syntax:** error = FmpRpProgram(fileDescriptor,rpName,  
options,error)

**rpName** Character string that either specifies or returns program name.

**options** Character string containing "C", "P", or both, to select either of the following options:

C clone – Create clone name if specified or assigned name is already assigned to an RP'd program. Program is not cloned if:

- System program has the assigned or specified name.
- Another program has the assigned or specified name, but it is not RP'd.
- No program with that name is currently RP'd.

P permanent – Do not release ID segment when program completes.

## FmpRunProgram

**Purpose:** Schedule a program.

**Syntax:** error = FmpRunProgram(string,params,runName)

**string** Character string specifying run string.

**params** Returns RMPAR parameters from program when it completes. If string specifies XQ, these parameters are meaningless.



## CI FILE HANDLING

**runName** Character string that returns true name used to schedule program.

### FmpRwBits

**Purpose:** Check string for letters R and W.

**Syntax:** value = FmpRwBits(string)

**string** Character string (up to 256 bytes).

**value** One of the following, depending upon string content:

- 0 neither present
- 1 W, but not R present
- 2 R, but not W present
- 3 R and W present

### FmpSetDcbInfo

**Purpose:** Change information in DCB.

**Syntax:** error = FmpSetDcbInfo(dcb,error,records,  
eofPos,recLen)

**records** Number of records in the file, plus one.

**eofPos** Current internal file position.

**recLen** Length of longest record, in words.

### FmpSetIoOptions

**Purpose:** Change I/O option word.

**Syntax:** error = FmpSetIoOptions(dcb,error,options)

**options** Same as for FmpIoOptions.

### FmpSetWord

**Purpose:** Change file position.

**Syntax:** error = FmpSetWord(dcb,error,position,how)

**position** Desired file position.

**how** Specifies whether file system should create extent to contain new position if it is outside the existing file area. 1 = extent creation is not permitted; 2 = extent creation is permitted.

## **FmpUniqueName**

**Purpose:** Create and return unique file name.

**Syntax:** Call FmpUniqueName(prefix,uniquename)

**prefix** Prefix for unique file name.

**uniquename**

Returns file name that is unique within system that contains no files from another system.

## **FmpWriteString**

**Purpose:** Write character string to file.

**Syntax:** length = FmpWriteString(dcb,error,string)

**length** Returns length of record written to file, or negative error code.

**string** Character string (up to 256 bytes) from which data is transferred.

## **SPECIAL-PURPOSE DS COMMUNICATION SUBROUTINES**

### **DsCloseCon**

**Purpose:** Close connection set up by DsOpenCon.

**Syntax:** error = DsCloseCon(conn)

**conn** Connection number of disc to be closed.

## CI FILE HANDLING

### DsDcbWord

**Purpose:** Return first word of DCB as it would appear if file associated with it were not opened via DS.

**Syntax:** error = DsDcbWord(conn,word)

**conn** Connection number of system.

**word** Returns first word of DCB.

### DsDiscInfo

**Purpose:** Return number of tracks and blocks per track for specified disc volume.

**Syntax:** error = DsDiscInfo(conn,lu,ntracks,bpert)

**conn** Connection number of system containing disc.

**lu** LU of disc volume.

**ntracks** Returns number of tracks on disc volume.

**bpert** Returns number of blocks per tracks on disc.

### DsDiscRead

**Purpose:** Read disc.

**Syntax:** error = DsDiscRead(conn,buf,len,track,sector)

**conn** Connection number of disc. Must have been set by DsSetDcbWord.

**buf** Buffer to hold data read from disc.

**len** Number of characters to read (up to 4096).

**track** Track from which to read.

**sector** 64-word sector from which to read (even number).

## DsFstat

**Purpose:** Perform FSTAT call for specified system.

**Syntax:** error = DsFstat(conn,buffer,lu,format,iop)

**conn** Connection number of system.

**buffer** At least 256 words.

**lu** LU of system on which to perform FSTAT.

**format** Same as for FSTAT.

**iop** Same as for FSTAT.

## DsOpenCon

**Purpose:** Open connection to remote user account/node.

**Syntax:** error = DsOpenCon(string,conn)

**string** Remote user account name, node name, or both, along with required delimiters. Must not contain a filename, only DS information.

**conn** Returns connection number.

## DsSetDcbWord

**Purpose:** Change first word of DCB so that DsDiscRead works.

**Syntax:** error = DsSetDcbWord(conn,word)

**conn** Connection number of disc to be read by DsDiscRead.

**word** Word to be changed.



# FMGR FILE HANDLING

CONTENT	PAGE
PARAMETERS .....	J-3
APOSN, EAPOS .....	J-3
CLOSE, ECLOS .....	J-3
CREAT, ECREA .....	J-3
CRETS .....	J-4
FCONT .....	J-4
FSTAT .....	J-5
IDCBS .....	J-6
LOCF, ELOCF .....	J-6
NAMF .....	J-6
OPEN, OPENF .....	J-7
POSTN, EAPOS .....	J-8
POST .....	J-8
PURGE .....	J-8
READF, EREAD .....	J-9
RWNDF .....	J-9
WRITF, EWRT .....	J-9

**PARAMETERS**

NOTE: The FMP calls beginning with E (eg. ECREA) can define larger files, up to 32767x128 blocks. The FMP calls not beginning with E (eg. CREAT) can only define files up to 16383 blocks, and 32767 records.

<b>IDCB</b>	A 144 word or longer, array used as the data control block (DCB).
<b>IERR</b>	Error return, see the negative FMGR error codes for meaning. If call is successful: OPEN,OPENF IERR= file type. CREAT IERR= number of sectors.
<b>INAM</b>	Six ASCII characters. First character not a blank or number, no embedded blanks, and (+, -) are not allowed. All six placed must be accounted for, and a Fortran DATA statement can be used to specify INAM.
<b>IBUF</b>	User buffer.
<b>ISC</b>	File security code: <0 read/write protected. =0 not protected (default). >0 write protected only.
<b>ICR</b>	Cartridge reference: >0 cartridge reference number. <0 logical unit number. =0 first one found (default). Order of search; private cartridges, then group cartridges, then system cartridges.
<b>IREC</b>	Next record number, double word for "E" type calls.
<b>IOFF</b>	Block offset of next record.
<b>IRB</b>	Relative block address of next record, double word for "E" type calls.
<b>IDCBS</b>	Actual size of DCB in words (only when IDCB > 144).

## APOSN AND EAPOS

CALL     APOSN  
          EAPOS     (IDCB,IERR,IREC<,<IRB<,<IOFF>>>)

Position a disc file (typically type 3) to a known record address. Record addresses are usually obtained through LOCF for APOSN, and ELOCF for EAPOS. IRB and IOFF are required for files with variable length records.

## CLOSE AND ECLOS

CALL     CLOSE  
          ECLOS     (IDCB<,<IERR>[,ITRUN])

Close DCB and make file available to others, can also truncate file size.

ITRUN     One word variable for CLOSE, double word variable for ECLOS.

+n number of blocks to be deleted from the end of the file when it is closed.

-n retain main file, delete extents.

0 standard close (default).

## CREAT AND ECREA

CALL     CREAT  
          ECREA     (IDCB,IERR,INAM,ISIZE,ITYPE  
                      [,ISC][,ICR][,IDCBS][,JSIZE])

Create a disc file.

IERR     Error return. If call is successful, IERR=number of sectors.

ISIZE     Two entry array describing file size: for CREAT a two word array, for ECREA a double word integer for each entry.

first entry — file size in blocks.

second entry — record length in words (used for type 2 files only).



## FMGR FILE HANDLING

**ITYPE** File type (1-32767).  
**JSIZE** Created file size in sectors; optional double word parameter returned by ECREA only.

### CRETS

CALL CRETS (IDCB,IERR,NUM,INAM  
[,ISIZE][,ITYPE][,ISC]  
[,ICR][,IDCBZ][,JSIZE])

CRETS creates a temporary or scratch disc file by making an entry in the File Directory and allocating disc space for the file. CRETS can define files up to 32767x128 blocks in size.

**NUM** Scratch file number, a one-word integer 0-99.  
**ISIZE** A double word integer for each entry.  
first entry — file size in blocks.  
second entry — record length in words (used for type 2 files only).  
**ITYPE** File type (1-32767).  
**JSIZE** Created file size in sectors; optional double word parameter returned if call was successful.

### FCONT

CALL FCONT(IDCB,IERR,ICON1<,ICON2>)

Control I/O functions on a non-disc type 0 file.

**ICON1** Control word, see EXEC 3 call for options.  
**ICON2** Additional control, see EXEC 3 call for options.

<b>FSTAT</b>
--------------

CALL FSTAT(ISTAT[,ILEN][,IFORM][,IOP][,IADD])

Return status of mounted cartridges.

ISTAT      Cartridge status buffer returned as FORMAT I or  
FORMAT II.

FORMAT I		
WORD	CONTENTS	CARTRIDGE
1	Logical Unit Number	First cartridge
2	Last FMP track	
3	Cartridge Reference Number	
4	Lock Word	
5	Logical Unit Number	Second cartridge
6	Last FMP track	
7	Cartridge Reference Number	
8	Lock Word	
9	Logical Unit Number	.
.	.	
.	.	
	0 no more discs	

where: Lock word is ID segment address of locking program or 0 (not locked).

FORMAT II		
WORD	CONTENTS	CARTRIDGE
1	Lock word      Logical unit #	First cartridge
2	Last FMP track	
3	Cartridge Reference Number	
4	ID	
5	Lock word      Logical unit #	Second cartridge
6	Last FMP track	
7	Cartridge Reference Number	
8	ID	
9	Lock word      Logical unit #	.
.	.	
.	.	
	0 no more discs	

where: Lock word is the offset of the ID segment in the Keyword Table or 0 (not locked).

ID identifies who mounted the cartridge.

## FMGR FILE HANDLING

ILEN	Length in words of status buffer (default= 125).
IFORM	Zero for FORMAT I. Non- zero for FORMAT II.
IOP	Type of cartridges to return information about: 1 = all cartridges mounted to the system. 0 = (under session) all private, group, and system cartridges mounted to that session. 0 = (non session) mounted system and non session cartridges.
IADD	0 if entire cartridge list was returned. Non-zero if entire cartridge list could not be returned.

### IDCBS

ISIZE=IDCBS(IDCBS)

Return actual DCB buffer area used (use only if IDCBS > 144).

### LOCF AND ELOCF

CALL      LOCFL  
          (I<sub>DCB</sub>, I<sub>ERR</sub>, I<sub>REC</sub> [, I<sub>RB</sub>] [, I<sub>OFF</sub>])  
          E<sub>LOCF</sub>      [, J<sub>SEC</sub>] [, J<sub>LU</sub>] [, J<sub>TY</sub>] [, J<sub>REC</sub>])

Retrieve status and location information from the data control block on an open file.

JSEC	File size in sectors; one word variable for LOCF, double word variable for ELOCF.
JLU	File lu.
JTY	File type.
JREC	Optional return for: record length (type 1 or 2 files). read/write code (type 0 files). meaningless (type 3 and above).

### NAMF

CALL NAMF(I<sub>DCB</sub>, I<sub>ERR</sub>, I<sub>NAM</sub>, M<sub>NAM</sub> [, I<sub>SC</sub>] [, I<sub>CR</sub>])

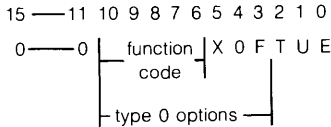
Close the DCB, if open, and rename file INAM to MNAM.

**OPEN AND OPENF**

CALL OPEN  
 OPENF (IDCB, IERR, INAM  
 [, IOPTN][, ISC][, ICR][, IDCBS])

Open a file for access.

- IERR Error return. If call is successful IERR=file type.
- INAM ASCII file name, or an integer containing a binary lu (OPENF only).
- IOPTN Open control word, defaults are:
  - exclusive use, only the calling program can access the file.
  - standard sequential output.
  - file type defined at creation is used for access.



- E bit 0 exclusive open;
  - 1 non exclusive open.
- U bit 0 non update open;
  - 1 update open.
- T bit 0 file type defined at creation (disc only);
  - 1 force file type to 1.
- F bit 0 use function code defined at creation (type 0 files only);
  - 1 use function code defined in bits 6-10 of IOPTN (for function codes see EXEC 3 call).
- X bit 0 file is not extendible (type 1 & 2 files only);
  - 1 file extents are to be created automatically when needed (type 1 & 2 files only).

**POSNT AND EAPOS**

POSNT  
 CALL (IDCB, IERR, NUR[, IR])  
 EPOSN

Position files relative to current file position or to a specific record number in any file type.

**NUR** Record position, a one word variable for POSNT or double word variable for EPOSN.

**IR** Position mode flag, the relationship between NUR and IR is:

NUR	IR = 0 OR OMITTED RELATIVE POSITION	IR ≠ 0 ABSOLUTE POSITION
NUR > 0	Position forward number of records specified	Position to record number specified
NUR = 0	No operation	No operation
NUR < 0	Position backward number of records specified	Error

**POST**

CALL POST(IDCB[, IERR])

Write contents of DCB to the disc, and save records in a file opened for non exclusive use. To lock the file for exclusive use with RNRQ call, use the following sequence:

1. call OPEN;
2. read file to pick up resource number;
3. call POST to clear DCB, no data is transferred;
4. call RNRQ to lock the file;
5. call READF to read the record to be modified;
6. modify the record and call WRITF to write it out;
7. call POST to transfer the updated record;
8. call RNRQ to unlock the file.

**PURGE**

CALL PURGE(IDCB, IERR, INAM<, ICS><, ICR>)

Delete named file INAM and all its extents, the file must not be open.

**READF AND EREAD**

CALL      READF  
             ( IDCB, IERR, IBUF[, IL][, LEN][, NUM])  
             EREAD

Read a record from an open file to the user buffer. If type 0 file, the number of words should be specified.

- IL**              Length of IBUF (read buffer), defaults are:  
                     file type = 0 zero length record.  
                     file type = 1 128 word record.  
                     file type > 1 actual record length.
- LEN**            Actual read length, set to -1 for EOF on sequential files only.
- NUM**            A one-word variable (for READF), or double-word variable (for EREAD) used to specify the record number to be read (default = start at current record number).

**RWPDF**

CALL RWPDF(IDCB[, IERR])

Rewind a magnetic tape or position a disc file to the first record in the file.

**WRITEF AND EWWRITE**

CALL      WRITEF  
             ( IDCB, IERR, IBUF[, IL][, NUM])  
             EWWRITE

Write a record from the user's buffer to an open file. For type 0 or type 3 and above, a specified number of words is written. For type 1 and 2 files the exact record length is written.

- IL**              Length of write buffer, defaults are:  
                     file type = 0 zero length record.  
                     file type = 1 128 word record.  
                     file type = 2 actual record length.  
                     file type > 2 zero length record.
- NUM**            Record number to be written. (default=start at current record number).



# GASP COMMANDS

CONTENT	PAGE
RU,GASP .....	K-2
AB .....	K-2
CJ .....	K-2
CS .....	K-3
DA .....	K-3
DJ .....	K-3
DS .....	K-4
EX .....	K-4
KS .....	K-4
RS .....	K-4
SD .....	K-5
SU .....	K-5
UP .....	K-5



## GASP

### **RU,GASP[,lu]**

Schedule GASP to prompt for command from lu (default=user's terminal).

### **RU,GASP,command**

Schedule GASP, execute command, then terminate.

- |                |  |
|----------------|--|
| <b>lu</b>      | Logical unit of interactive device on which GASP commands are entered. In a session environment lu must be specified if it is different from the session logical unit. |
| <b>command</b> | Any GASP operator command.   |

### **^AB,job # ,[u.g]**

Before a job is processed, it may be removed with the AB command.

- |              |  |
|--------------|--|
| <b>job #</b> | Number assigned to job by spool system; use DJ to display job numbers. |
| <b>u.g</b>   | Aborts all jobs owned by session account (user.-group).                |

### **^CJ,job # < ,H ,R >**

Change job priority or status. Only used for a job in I, R, or RH status.

- |                 |  |
|-----------------|--|
| <b>job #</b>    | Number assigned to job by spool system; use DJ to display job numbers. |
| <b>priority</b> | New job priority; only allowed before job is active.                   |
| <b>H</b>        | Hold job from processing; changes R status to RH, and I to IH.         |
| <b>R</b>        | Release job for processing; changes RH status to R.                    |

**^CS,spoolfile**<,<sup>priority</sup>H,R>

Change status of outspool file or change spool priority if outspool file is not active.

<b>spoolfile</b>	Name of spool file as displayed by DJ.
<b>priority</b>	New outspool priority.
<b>H</b>	Hold spool file; if active, changes status to AH; if waiting, changes status to H.
<b>R</b>	Release spool file that has been held in AH or H status.

**^DA**

Deallocate spooling. Before using DA, the spool system must be shut down, all files must be closed, and all current job processing and/or outspooling should be completed.

Only the system manager can execute this command.

Response:

**KILL SPOOLING?** The system prints this message in response to DA in order to give you a chance to change your mind.

**^DJ[AL]**<<sup>job #</sup>or [u.g]>  
jobname




Display the job number, job name, job status, priority, user.group, and the spool pool files assigned to the job except for the job input spool.

<b>AL</b>	Causes all jobs (session and non-session) to be reported.
<b>job #</b>	Job number of particular job to be displayed.
<b>jobname</b>	Name of the job or jobs to be displayed. If both <b>job #</b> and <b>jobname</b> are omitted, all jobs currently in the system for the current user are displayed.
<b>u.g</b>	Reports only jobs belonging to the user.group account of u.g. If the '@' character is used for either the user or group, then all session users or groups (or both) are reported.

## GASP

### **^DS**[AL][,lu[,u.g]]

Display the spool file name, job number, user.group name, outspool priority, spool status, and the logical unit to which the file is being or will be outspooled.


- |            |   |   |
|------------|---|---|
| <b>AL</b>  | Causes all spools (session and non-session) to be reported.   |  |
| <b>lu</b>  | Outspool logical unit; only files directed to this lu are displayed; if omitted, all files in the outspool queue are displayed. If in session, lu is the session lu, and the lu displayed is the system lu that the session lu maps to. |  |
| <b>u.g</b> | Reports only files belonging to the account of u.g. If the '@' character is used for either the user or group, then all users or groups (or both) are reported.   |  |

### **^EX**

Terminate GASP.



**^KS**< ,spoolfile  
,lu > [,u.g]

Remove outspool file from the outspool queue.

- |                  |  |   |
|------------------|--|---|
| <b>spoolfile</b> | Name of spool file to be removed.  |   |
| <b>lu</b>        | Logical unit of device to which file is being outspooled. When running under session, lu is the session logical unit number. |   |
| <b>u.g</b>       | Kills all spool files owned by session account u.g.  |  |

### **^RS**,spoolfile[,lu]

Restart active outspool file from the beginning.

- |                  |  |   |
|------------------|--|---|
| <b>spoolfile</b> | Name of active or active-held spool file in outspool queue.  |  |
| <b>lu</b>        | New logical unit to which file is to be outspooled; if omitted, logical unit previously assigned is used for spool output. |  |

**^SD**<B[ATCH],S[POOL]>

Hold all spooled jobs, all spooled output, or both.

- |             |   |
|-------------|---|
| <b>B</b>    | Hold all pending jobs: spool files are not affected.  |
| <b>S</b>    | Hold all pending spool files; job processing is not affected.   |
| <b>none</b> | If both B and S are omitted, then both job processing and outspooling are held. Inspooling by JOB may continue. |

**^SU**<B[ATCH],S[POOL]>

Start up spool system after it has been shut down with SD.

- |             |   |
|-------------|---|
| <b>B</b>    | Jobs held with SD are released; does not restart outspooling.         |
| <b>S</b>    | Outspools held with SD are released; does not restart job processing. |
| <b>none</b> | Both jobs and outspools held by SD are restarted.                     |

**^UP**[,RS]

Up outspool device.

- |           |  |
|-----------|--|
| <b>RS</b> | Restart active files from the beginning. |
|-----------|--|



# ACCOUNT COMMANDS

CONTENT	PAGE
EX .....	K-8
HE .....	K-8
LI .....	K-8
TE .....	K-8
/A .....	K-8
TR .....	K-8
/E .....	K-8

## ACCOUNT

### ACCOUNT ID FORMAT

#### USER.GROUP

@."group" — All users in group.

"user".@ — All users named "USER".

@.@ — All users.

#### EX[IT]

Terminate the account program.

#### HE[LP][,keyword[,list]]

List valid commands and scheduled HELP utility.

#### LI[ST],A[CCT][,<list namr>]

List session wide information.

#### LI[ST],G[ROUP],<group>[,<list namr>]

List one or more group account entries.

#### LI[ST],U[SER],<user.group>[,<list namr>]

Lists one or more user account entries.

#### TE[LL],<user.group>[<,namr>][,<MESSAGE>]

Send a message to a single active user or group, or to all active sessions.

#### /A

Abort current command.

#### TR [,control[,list<[NO[ECHO]]] [EC[HO]]]]

Invoke a transfer from within a command.

#### /E

End current phase.

# BATCH AND SPOOLING COMMANDS

CONTENT	PAGE
AB .....	K-10
CS .....	K-10
EOJ .....	K-10
JOB .....	K-11
SL .....	K-11
RUN .....	K-12
TL .....	K-12
XE .....	K-13



## BATCH AND SPOOLING

### AB

30

Terminate batch job.

### CS,lu,attribute

30

Modify or change spool options set up by SL command.

lu	lu defined at set up.
attribute	one of the following:
RWind	reset file to first record.
PUrge	change SAve flag to PUrge.
SAve	change PUrge flag to SAve.
PAss	remove HOld option.
ENd	write EOF and terminate spool. Spool file placed in outspool queue (default).
BUFFER	change to buffering.
NBUFFER	change to no buffering.
NPass	change lu and/or priority information, by specifying the 2 additional parameters: [.outlu[,priority]] outlu = new lu priority = new priority

### EOJ[,RP[,RG]]

30

End of spooled job.

RP	Dismount job's private session cartridges. (Default=leave mounted.)
RG	Dismount job's group session cartridges. (Default=leave mounted.)

**JOB**[,name[:hr:min:sec][,user[,priority[,spool  
priority][,sp]]]]

30

Initiate job for spooling.

- name** Job name.
- :hr:min:sec** CPU time limit for job in hours, minutes, seconds.
- user** Session user account ID in the form "user.group/  
password". If a job is submitted outside of a session  
when session is installed this parameter must be  
specified.
- priority** Job priority in range from 1-255 (default = 99).
- spool  
priority** Outspool priority (default=priority).
- sp** Specify:
  - NO Outspool now,or
  - NS No outspooling.

**SL**,lu[,filedes[,attribute[,outlu[,priority  
[,prog]]]]]]

30/50

Spool setup and outspool control.

- lu** The session lu to which a spool file is to be as-  
sociated. The lu must not be LU2 (system disc), LU3  
(auxiliary disc), any lu associated with a disc driver,  
a spool lu, or if in a job system LU5 (standard spool  
input device).
- filedes** name of existing file to be used as a spool file (de-  
fault=system assigns spool pool file).
- attribute** defines characteristics of spool access. Any 3 attri-  
bute codes can be combined, no delimiters  
necessary.  
  
attribute codes:
  - NO = Queue file for immediate outspool
  - RE = Read only
  - WR = Write only
  - BO = Both read and write
  - WN = Write now
  - BU = Buffered
  - PU = Purge
  - SH = Write spool headers
  - ST = Standard file codes:

## BATCH AND SPOOLING

default for attribute codes:

	outlu specified	outlu not specified	
filedes specified	WRITE,HOLD, SPOOL HEADERS, SAVE	READ,HOLD, STANDARD FORMAT, SAVE	
filedes not specified	WRITE,HOLD, SPOOL HEADERS, SPOOL POOL FILE,PURGE	BOTH,HOLD, STANDARD FORMAT, SPOOL POOL FILE,PURGE	

**priority** Outspool priority (default=session-99, Batch-priority of job).

**prog** If specified, program "prog" will be scheduled, with wait, by the spool system when spool lu is closed. Note the spool file will not be outspooled, "prog" must properly dispose of the file. Required capability of 50.

**outlu** Session lu for outspooling.

### **RUN,JOB,namr [,priority]**

30

Run batch job.

**namr** File name of file containing single job to be spooled, or logical unit of input device containing jobs to be spooled; (default=session terminal, or logical unit 5 if outside of session).

**priority** Priority of job (default=99).

### **TL:hr:min:sec**

30

Set run time limit.

**:hr:min:sec** Time limit for execution of any programs with RU command subsequent to TL command. If omitted, job time limit is used.

**XE,namr[,priority]**

Job input control.

**namr** Identifies input device containing a job to be placed in job queue, may be a logical unit or the name of an existing file.

**priority** Job priority (default=99).



# SMP CALLS

CONTENT	PAGE
PARAMETERS .....	L-2
SPOPN .....	L-2
WORKING CALLS .....	L-3
RETRIEVE RECORD POSITION .....	L-3
CHANGE RECORD POSITION .....	L-3

## SMP CALLS

### PARAMETERS

- ISMP 3 word array containing name of program SMP.
- ISLU Spool lu returned by SPOP call. Each subsequent spool call must specify this lu.

### SPOP

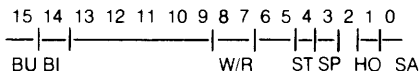
#### CALL SPOP(IBUFR,ISLU)

Make a spool file active and ready for use.

- IBUFR 16 word set up buffer structured as follows:

word contents

- 0 =0 if no batch input checking desired.
- 1 >0 session lu for the spool file; or  
=0 SMP allocates a session lu for the spool file; or  
=1 a direct map to system lu is set up.
- 5 security code.
- 6 cartridge reference number.
- 7 driver type, in octal.
- 8 disposition flags:



- BU 1= buffered; 0= not buffered.
- BI 1= batch input; 0 otherwise;
- W/R 10B= write; 01B= read; 00B= write/  
read.
- ST 1= standard file; 0= spool file.
- SP 1= spool pool file; 0= user file.
- HO 1= hold outspool; 0= outspool now.
- SA 1= save file; 0= purge.

- 9 spool priority (1-9999).
- 10 spool status (used by SMP,GASP).
- 11 if batch — job number; if not batch — directory entry number of session program.
- 12-14 set to 0 or program parameter of SL command.
- 15 outspool lu.

- ISLU Spool lu return.

## WORKING CALLS

### CALL EXEC(23,ISMP,XX,ISLU)

- XX
- =1 Change purge to save.
  - =2 Change save to purge.
  - =3 Queue for outspooling.
  - =4 EOF and queue for outspooling.
  - =5 Change spool options; use additional parameters NOL and NPR following ISLU for this call only.
    - NOL new outspool lu (default=previous lu).
    - NPR new outspool priority (default=previous value).
  - =6 Set buffer flag.
  - =7 Clear buffer flag.

## RETRIEVE RECORD POSITION

### CALL EXEC(23,ISMP,8,ISLU)

CALL RMPAR(IPRM) — for parameter pick up.

- IPRM
- 5 word array containing pointers to record position.
    - word 1 =
    - word 2 =
    - word 3 =
- } contain an internal coding of the current position of the referenced file.
- word 4 = not used but should be included in array.
  - word 5 = not used but should be included in array.

## CHANGE RECORD POSITION

### CALL EXEC(23,ISMP,9,ISLU,IPRM1,IPRM2,IPRM3)

- IPRM1-3 Record position from the RETRIEVE RECORD call.





# VMA/EMA ROUTINES

CONTENT	PAGE
GENERAL PURPOSE VMA/EMA SUBROUTINES .....	L-6
EMAST .....	L-6
VMAST .....	L-6
VMAIO .....	L-6
EIOSZ .....	L-7
LKEMA/ULEMA .....	L-7
FMGR VMA FILE ROUTINES .....	L-8
PARAMETERS .....	L-8
CLSVM .....	L-8
CREVM .....	L-9
OPNVM .....	L-9
PSTVM .....	L-9
PURVM .....	L-9
VREAD .....	L-10
VWRIT .....	L-10
VMA/EMA MAPPING MANAGEMENT SUBROUTINES .....	L-11
.IMAP SUBROUTINE .....	L-11
.IRES SUBROUTINE .....	L-11
.JMAP SUBROUTINE .....	L-12
.JRES SUBROUTINE .....	L-12
.MMAP SUBROUTINE .....	L-13
.ESEG SUBROUTINE .....	L-13
.LBP, .LBPR SUBROUTINE .....	L-14
.LPX, .LPXR SUBROUTINE .....	L-14
.EMIO SUBROUTINE .....	L-15

## VMA/EMA ROUTINES

### GENERAL PURPOSE VMA/EMA SUBROUTINES

#### EMAST

**Purpose:** Return information about VMA or EMA.

**Syntax:** CALL EMAST (nema,nmseg,imseg[,iws])

**nema** Total page size of VMA or EMA (not including page table).

**nmseg** Total page size of mapping segment (MSEG).

**imseg** Starting logical page of MSEG.

**iws** Working set page size. For EMA, same as nema.

**Upon return:**

A-register = 0 if normal return  
              = -1 if error occurred

#### VMAST

**Purpose:** Return size of VMA or EMA.

**Syntax:** CALL VMAST (ivma, isize)

**ivma** -2 = Not VMA or EMA program, isize = 0  
          0 = EMA program  
          1 = VMA program

**isize** VMA or EMA page size.

#### VMAIO

**Purpose:** Perform large VMA or EMA data transfers to or from device.

Syntax: CALL VMAIO (ecode,cntrl,ibuff,ilen[,param3[,param4]])

**ecode** 1 for read, 2 for write.

**cntrl** Two-word quantity with the following format:

bits 0-5	Device LU
bits 6-15	Reserved
bits 0-5	Reserved
bits 6-15	Same as EXEC(1,2) cntwd

LU Number (bits 0-5) is LU of device that data is to be transferred to or from.

Bits 6-15 of word 2 are identical to bits 6-15 of the EXEC(1,2) cntwd.

**ibuff** VMA/EMA word offset to start of buffer (two-word integer).

**ilen** Length of ibuff (one-word integer); negative number of characters or positive number of words.

**param3** Optional parameter or buffer, as in EXEC 1 or 2 call.

**param4** Optional parameter or optional buffer length.

**EIOSZ**

**Purpose:** Determine maximum length of transfer.

**Syntax:** CALL EIOSZ (isize)

isize = EIOSZ ( )

isize Always returns 100000B.

**LKEMA/ULEMA**

**Purpose:** Lock/unlock a shareable EMA partition.

**Syntax:** CALL LKEMA (lock)

CALL ULEMA (unlock)

## VMA/EMA ROUTINES

### FMGR VMA FILE ROUTINES

#### PARAMETERS

- IDCB** A 144 word or longer array used as the data control block (DCB).
- IERR** Error return, see the negative FMP error codes for meaning. If successful:  
OPNVM IERR=file type
- INAM** Six ASCII characters. First character not a blank or number, no embedded blanks, and (+, -;) are not allowed. All six places must be accounted for, and a FORTRAN DATA statement can be used to specify INAM.
- ISC** File security code:  
<0 read/write protected.  
=0 not protected (default).  
>0 write protected only.
- ICR** Cartridge reference:  
>0 cartridge reference number.  
<0 logical unit number.  
=0 first one found (default). Order of search; private cartridges, then group cartridge, then system cartridges.

#### CLSVM

- Purpose:** Post all the pages of the working set to the VMA backing store file and execute an FMP close on the VMA backing store file.
- Syntax:** CALL CLSVM

**CREVM**

**Purpose:** Create the VMA backing store file.

**Syntax:** CALL CREVM ([,INAM][,IERR][,IOPTN][,ISC][,ICR])

**IOPTN** File options:

Bit 0 = 1 Create a non-scratch file (file INAM) to be used as the backing store file.

Bit 1 = 1 Open file if create fails.

Bit 2 = 1 File create is deferred until required.

Bit 3 = 1 File extents are not created.

**OPNVM**

**Purpose:** Open the VMA backing store file.

**Syntax:** CALL OPNVM (INAM[,IERR][,IOPTN][,ISC][,ICR])

**IOPTN** File options:

Bit 0 = 1 Open file for non-exclusive use.

Bit 1 = 1 Open file for update (initialized backing store file).

Bit 2 = 1 File open is deferred until required.

Bit 3 = 1 File extents are not to be created.

Bit 4 = 1 Read-only access.

**PSTVN**

**Purpose:** Post all the pages in the working set to the VMA backing store file.

**Syntax:** CALL PSTVM

**PURVM**

**Purpose:** Purge the VMA backing store file.

**Syntax:** CALL PURVM

## VMA/EMA ROUTINES

### VREAD

**Purpose:** Read records from a file into a VMA or EMA array.

**Syntax:** CALL VREAD (IDCB,IERR,IARRAY,IDL[,ILEN][,INUM])

**IERR** Error return values:  
0 = normal return  
<0 = FMP error  
1 = request parameter error  
2 = VMA/EMA mapping error

**IARRAY** Data transfer destination start address in VMA or EMA (two word address set up by the compiler).

**IDL** Data length (positive number of words).

**ILEN** Data length read (ILEN=-1 for EOR).

**INUM** Record number to read (default=current record).

### VWRITE

**Purpose:** Write records from a VMA or EMA array into a file.

**Syntax:** CALL VWRITE (IDCB,IERR,IARRAY,IDL [,ILEN][,INUM])

**IERR** Error return values:  
0 = normal return  
<0 = FMP error  
1 = request parameter error  
2 = VMA/EMA mapping error

**IARRAY** Data transfer destination start address in VMA or EMA (two-word address set up by the compiler).

**IDL** Data length requested (positive number of words).

**INUM** Record number to write (default=current record).

## VMA/EMA MAPPING MANAGEMENT SUBROUTINES

### .IMAP SUBROUTINE

**Purpose:** Resolve address of array element with one-word integer subscripts and map it into logical memory.

**Macro/1000 calling sequence:**

```
EXT .IMAP
JSB .IMAP
DEF TABLE Address of table containing array parameters.
DEF An      Address of nth subscript value.
      :
      :
DEF A1      Address of 1st subscript value.
```

**RTN** normal return

**Normal return:** B-Register contains logical address of element.  
A-Register is undefined.

### .IRES SUBROUTINE

**Purpose:** Resolve address of array element with one-word integer subscripts.

**Macro/1000 calling sequence:**

```
EXT .IRES
JSB .IRES
DEF TABLE Address of table containing array parameters.
DEF An      Address of nth subscript value.
      :
      :
DEF A1      Address of 1st subscript value.
```

**RTN** normal return

**Normal return:** A- and B-Registers contain offset of array element into EMA or VMA in double integer format (most significant word in A-Register, least significant word in B-Register).



## VMA/EMA ROUTINES

### .JMAP SUBROUTINE

**Purpose:** Resolve address of array element with double integer subscripts and map it into logical memory.

Macro/1000 calling sequence:

```
EXT .JMAP
JSB .JMAP
DEF TABLE Address of the array description table.
DEF An      Address of nth subscript value.
DEF A(n-1) Address of (n-1) subscript value.
:           :
DEF A1      Address of 1st subscript value.
```

RTN normal return

Normal return: VMA or EMA array element resides in physical memory, last two user map registers (VSEG) point to that element, and B-Register contains logical address of element. A-Register is undefined.

### .JRES SUBROUTINE

**Purpose:** Resolve address of array element with double integer subscripts.

Macro/1000 calling sequence:

```
EXT .JRES
JSB .JRES
DEF TABLE Address of table containing array parameters.
DEF An      Address of nth subscript value.
:           :
DEF A1      Address of 1st subscript value.
```

RTN normal return

Normal return: A- and B-Registers contain offset of array element into VMA or EMA in double integer format (most significant word in A-Register, least significant word in B-Register).

**MMAP SUBROUTINE**

**Purpose:** Map consecutive pages of EMA or VMA into logical memory.

FORTTRAN calling sequence:

CALL MMAP(ipgs,npgs)

**ipgs** VMA/EMA page holding start of buffer to map  
(where first page in EMA or VMA is page 0).

**npgs** Number of pages (rounded up) in buffer to be mapped.

Macro/1000 calling sequence:

```
EXT MMAP
JSB MMAP
DEF RTN
DEF IPGS
DEF NPGS
```

RTN return point

Upon return:

**A-Register** = 0 if normal return  
= -1 if an error occurred.

**.ESEG SUBROUTINE**

**Purpose:** Map several pages of EMA or VMA (not necessarily contiguous) into logical memory.

Macro/1000 calling sequence:

```
EXT .ESEG
:
LDB number Number of map registers to modify.
JSB .ESEG
DEF *+2 Error return point (not used).
DEF PBUFR Table of pages to map.
```

RTN error return (Not used.)  
RTN+1 normal return normal return point.

## VMA/EMA ROUTINES

Normal return: All VMA/EMA pages are mapped into logical memory and B-Register = logical address of the starting page of MSEG.

### **.LBP, .LBPR SUBROUTINE**

Purpose: Convert virtual address to logical address.

Macro/1000 calling sequence:

EXT .LBP		EXT .LBPR
DLD PONTR	or	JSB .LBPR
JSB .LBP		DEF PONTR

where:

PONTR Double integer pointer (high word first) containing virtual address.

Normal return: B-Register contains logical address; A-Register contains data's page number in physical memory.

### **.LPX, .LPXR SUBROUTINE**

Purpose: Convert virtual address of offset to logical address.

Macro/1000 calling sequence:

EXT .LPX		EXT .LPXR
DLD PONTR	or	JSB .LPXR
JSB .LPX		DEF PONTR
DEF OFSET		DEF OFSET

where:

PONTR Double integer pointer containing the virtual address.

OFSET Double integer offset from the virtual address.

Normal return: B-Register contains logical address; A-Register contains data's page number in physical memory.

**.EMIO SUBROUTINE**

**Purpose:** Map in up to MSEG-size buffer, which can then be used for I/O.

Macro/1000 calling sequence:

```

EXT .EMIO
JSB .EMIO
DEF RTN      Address for error return
DEF BUFL     Number of words in the buffer
DEF TABLE  Table containing array parameters
DEF An      Subscript value for nth dimension
:           :
DEF A1      Subscript value for 1st dimension

```

RTN error return  
normal return

Normal return: B-Register contains logical address of element.  
A-Register is meaningless.

Error return: At location RTN. A-Register contains "16" (ASCII);  
B-Register contains "EM" (ASCII).



# TABLES

CONTENT	PAGE
ASCII/BYTES .....	M-2
ASCII CHARACTERS AND BINARY CODES .....	M-3
RTE SPECIAL CHARACTERS .....	M-4
INSTRUCTION CODES IN OCTAL .....	M-4
BASE SET INSTRUCTION CODES IN BINARY .....	M-6
EXTENDED INSTRUCTION GROUP CODES .....	M-8
SYSTEM COMMUNICATION AREA LOCATIONS .....	M-11
DEVICE REFERENCE TABLE (DRT) .....	M-15
EQUIPMENT TABLE (EQT) .....	M-15
DEVICE STATUS TABLE .....	M-18
DEVICE FILE DCB .....	M-23
EQT WORD 6 .....	M-24
ID SEGMENT .....	M-25
ID SEGMENT EXTENSIONS .....	M-27
SESSION CONTROL BLOCK (SCB) .....	M-28
SYSTEM DISC LAYOUT .....	M-29
DATA CONTROL BLOCK (DCB) .....	M-30
CARTRIDGE DIRECTORY FORMAT .....	M-33
FILE DIRECTORY .....	M-34
DISC DIRECTORY, FILE ENTRY .....	M-35
DISC DIRECTORY, TYPE 0 FILE ENTRY .....	M-36
DISC FILE RECORD FORMATS .....	M-37
TYPE 6 FILE FORMAT .....	M-38
DISC VOLUME HEADER FORMAT .....	M-39
DIRECTORY STRUCTURE .....	M-40
ROOT DIRECTORY HEADER/TRAILER .....	M-41
ROOT DIRECTORY ENTRY .....	M-41
DIRECTORY HEADER/TRAILER FORMAT .....	M-42
FILE ENTRY .....	M-42
SUBDIRECTORY ENTRY .....	M-43
EXTENT ENTRY .....	M-43
RECORD FORMATS .....	M-44
ABSOLETE TAPE FORMAT .....	M-61
FMGR GLOBAL EQUIVALENCE TABLE .....	M-62
GENERAL WAIT STATE MESSAGES .....	M-63

## ASCII/BYTES

BYTE POSITION			
CHAR	Left	Right	Dec.
A	040400	000101	65
B	041000	000102	66
C	041400	000103	67
D	042000	000104	68
E	042400	000105	69
F	043000	000106	70
G	043400	000107	71
H	044000	000110	72
I	044400	000111	73
J	045000	000112	74
K	045400	000113	75
L	046000	000114	76
M	046400	000115	77
N	047000	000116	78
O	047400	000117	79
P	050000	000120	80
Q	050400	000121	81
R	051000	000122	82
S	051400	000123	83
T	052000	000124	84
U	052400	000125	85
V	053000	000126	86
W	053400	000127	87
X	054000	000130	88
Y	054400	000131	89
Z	055000	000132	90
a	060400	000141	97
b	061000	000142	98
c	061400	000143	99
d	062000	000144	100
e	062400	000145	101
f	063000	000146	102
g	063400	000147	103
h	064000	000150	104
i	064400	000151	105
j	065000	000152	106
k	065400	000153	107
l	066000	000154	108
m	066400	000155	109
n	067000	000156	110
o	067400	000157	111
p	070000	000160	112
q	070400	000161	113
r	071000	000162	114
s	071400	000163	115
t	072000	000164	116
u	072400	000165	117
v	073000	000166	118
w	073400	000167	119
x	074000	000170	120
y	074400	000171	121
z	075000	000172	122
0	030000	000060	48
1	030400	000061	49
2	031000	000062	50
3	031400	000063	51
4	032000	000064	52
5	032400	000065	53
6	033000	000066	54
7	033400	000067	55
8	034000	000070	56
9	034400	000071	57

BYTE POSITION			
CHAR	Left	Right	Dec.
NUL	000000	000000	0
SOH	000400	000001	1
STX	001000	000002	2
ETX	001400	000003	3
EOT	002000	000004	4
ENO	002400	000005	5
ACK	003000	000006	6
BEL	003400	000007	7
BS	004000	000010	8
HT	004400	000011	9
LF	005000	000012	10
VT	005400	000013	11
FF	006000	000014	12
CR	006400	000015	13
SO	007000	000016	14
SI	007400	000017	15
DLE	010000	000020	16
DC1	010400	000021	17
DC2	011000	000022	18
DC3	011400	000023	19
DC4	012000	000024	20
NAK	012400	000025	21
SYN	013000	000026	22
ETB	013400	000027	23
CAN	014000	000030	24
EM	014400	000031	25
SUB	015000	000032	26
ESC	015400	000033	27
FS	016000	000034	28
GS	016400	000035	29
RS	017000	000036	30
US	017400	000037	31
SPACE	020000	000040	32
!	020400	000041	33
"	021000	000042	34
=	021400	000043	35
\$	022000	000044	36
%	022400	000045	37
&	023000	000046	38
'	023400	000047	39
(	024000	000050	40
)	024400	000051	41
*	025000	000052	42
+	025400	000053	43
,	026000	000054	44
-	026400	000055	45
.	027000	000056	46
/	027400	000057	47
:	035000	000072	58
;	035400	000073	59
<	036000	000074	60
=	036400	000075	61
>	037000	000076	62
?	037400	000077	63
@	040000	000100	64
[	055400	000133	91
]	056000	000134	92
^	056400	000135	93
_	057000	000136	94
`	057400	000137	95
~	060000	000140	96
	075400	000173	123
	076000	000174	124
	076400	000175	125
	077000	000176	126
DEL	077400	000177	127

# ASCII CHARACTERS AND BINARY CODES

BITS		COLUMN	0 <sub>00</sub>	0 <sub>01</sub>	0 <sub>10</sub>	0 <sub>11</sub>	1 <sub>00</sub>	1 <sub>01</sub>	1 <sub>10</sub>	1 <sub>11</sub>
b <sub>7</sub>	b <sub>6</sub> b <sub>5</sub>	ROW ↓	0	1	2	3	4	5	6	7
0	0 0 0 0	0	NUL	DLE	SP	0	⊕	P	.	p
0	0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0	0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0	0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0	0 1 0 0	4	EOT	DC4	\$	4	D	T	d	t
0	0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0	0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0	0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1	0 0 0 0	8	BS	CAN	(	8	H	X	h	x
1	0 0 0 1	9	HT	EM	)	9	I	Y	i	y
1	0 0 1 0	10	LF	SUB	*	.	J	Z	j	z
1	0 0 1 1	11	VT	ESC	+	:	K	[	k	{
1	0 1 0 0	12	FF	FS	,	<	L	\	l	
1	0 1 0 1	13	CR	GS	-	=	M	]	m	}
1	0 1 1 0	14	SO	RS	.	>	N	^	n	~
1	0 1 1 1	15	SI	US	/	?	O	_	o	DEL

32 CONTROL CODES

Upshifted Lower Case

64 CHARACTER SET

96 CHARACTER SET

128 CHARACTER SET

EXAMPLE: The representation for the character "K" (column 4, row 11) is

	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>
BINARY	1	0	0	1	0	1	1
OCTAL	1	1	3				

\* Depressing the Control key while typing an upper case letter produces the corresponding control code on most terminals. For example, Control-H is a backspace.



## RTE SPECIAL CHARACTERS

Mnemonic	Octal Value	Use
SOH (Control A)	1	Backspace (TTY)
EM (Control Y)	31	Backspace (2600)
BS (Control H)	10	Backspace (TTY, 2615, 2640, 2644, 2645)
EOT (Control D)	4	End-of-file (TTY 2615, 2640, 2644, 2645)

## INSTRUCTION CODES IN OCTAL

Memory Reference		Ext. Inst. Group
ADA 04(0XX)...	CMA 003000	ADX 105746
ADB 04(1XX)...	CMB 007000	ADY 105756
AND 01(0XX)...	CME 002200	CAX 101741
CPA 05(0XX)...	INA 002004	CAY 101751
CPB 05(1XX)...	INB 006004	CBS 105774
IOR 03(0XX)...	RSS 002001	CBT 105766
ISZ 03(1XX)...	SEZ 002040	CBX 105741
JMP 02(1XX)...	SLA 002010	CBY 105751
JSB 01(1XX)...	SLB 006010	CMW 105776
LDA 06(0XX)...	SSA 002020	CXA 101744
LDB 06(1XX)...	SSB 006020	CXB 105744
STA 07(0XX)...	SZA 002002	CYA 101754
STB 07(1XX)...	SZB 006002	CYB 105754
XOR 02(0XX)...		DSX 105761
↑ Binary		DSY 105771
	<b>Input/Output</b>	ISX 105760
	CLC 1067..	ISY 105770
	CLF 1031..	JLY 105762
	CLO 103101	JPY 105772
	HLT 1020..	LAX 101742
	LIA 1025..	LAY 101752
	LIB 1065..	LBT 105763
	MIA 1024..	LBX 105742
	MIB 1064..	LBY 105752
	OTA 1026..	LDX 105745
	OTB 1066..	LDY 105755
	SFC 1022..	MBT 105765
	SFS 1023..	MVW 105777
	SOC 102201	SAX 101740
	SOS 102301	SAY 101750
	STC 1027..	SBS 105773
	STF 1021..	SBT 105764
	STO 102101	SBX 105740
		SBY 105750
	<b>Extended Arithmetic</b>	SFB 105767
	ASL 1000(01X)-	STX 105743
	ASR 1010(01X)-	STY 105753
	DIV 100400	TBS 105775
	DLD 104200	XAX 101747
	DST 104400	XAY 101757
	LSL 1000(10X)-	XBX 105747
	LSR 1010(10X)-	XBY 105757
	MPY 100200	
	RRL 1001(00X)-	
	RRR 1011(00X)-	
	↑ Binary	
<b>Shift-Rotate</b>		
ALF 001700		
ALR 001400		
ALS 001000		
ARS 001100		
BLF 005700		
BLR 005400		
BLS 005000		
BRS 005100		
CLE 000040		
ELA 001600		
ELB 005600		
ERA 001500		
ERB 005500		
NOP 000000		
RAL 001200		
RAR 001300		
RBL 005200		
RBR 005300		
SLA 000010		
SLB 004010		
<b>Alter-Skip</b>		
CCA 003400		
CCB 007400		
CCE 002300		
CLA 002400		
CLB 006400		
CLE 002100		

# INSTRUCTION CODES IN OCTAL (CONTINUED)

Floating Point	Fast FORTRAN	Dynamic Mapping System
FAD 105000	DBLE 105201	DJP 105732
FDV 105060	DDINT 105217	DJS 105733
FIX 105100	SNGL 105202	JRS 105715
FLT 105120	BLE 105207	LFA 101727
FMP 105040	CFER 105231	LFB 105727
FSB 105020	DFER 105205	MBF 105703
FIXD 105104	ENTP 105224	MBI 105702
FLTD 105124	ENTR 105223	MBW 105704
TADD 105002	FLUN 105226	MWF 105706
TDIV 105062	GOTO 105221	MWI 105705
.TFTD 105126	.NGL 105214	MWW 105707
TFTS 105122	PACK 105230	PAA 101712
TFXD 105106	.PWR2 105225	PAB 105712
TFXS 105102	\$SETP 105227	PBA 101713
TMPY 105042	.XCOM 105215	PBB 105713
TSUB 105022	.XFER 105220	RSA 101730
XADD 105001	.XPAK 105206	RSB 105730
.XDIV 105061	.DCM 105216	RVA 101731
XFTD 105125	.FCM 105232	RVB 105731
XFTS 105121	.MAP 105222	SJP 105734
.XFXD 105105	.TCM 105233	SJS 105735
XFXS 105101		SSM 105714
XMPY 105041	<b>Double Integer</b>	SYA 101710
XSUB 105021	.DAD 105014	SYB 105710
	.DCO 105204	UJP 105736
	.DDE 105211	UJS 105737
	.DDI 105074	USA 101711
	.DDIR 105134	USB 105711
	.DDS 105213	XCA 101726
	.DIN 105210	XCB 105726
	.DIS 105212	XLA 101724
	.DMP 105054	XLB 105724
	.DNG 105203	XMA 101722
	.DSB 105034	XMB 105722
	.DSBR 105114	XMM 105720
		XMS 105721
		XSA 101725
		XSB 105725
<b>Scientific Inst. Set</b>		
ALOG 105322		
ALOGT 105327		
ATAN 105323		
COS 105324		
EXP 105326		
SIN 105325		
SQRT 105321		
TAN 105320		
TANH 105330		
DPOLY 105331		
.CMRT 105332		
ATLG 105333		
FPWR 105334		
TPWR 105335		

## BASE SET INSTRUCTION CODES IN BINARY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	AND	001		0	Z/C										
D/I	XOR	010		0	Z/C										
D/I	IOR	011		0	Z/C										
D/I	JSB	001		1	Z/C										
D/I	JMP	010		1	Z/C										
D/I	ISZ	011		1	Z/C										
D/I	AD*	100		A/B	Z/C										
D/I	CP*	101		A/B	Z/C										
D/I	LD*	110		A/B	Z/C										
D/I	ST*	111		A/B	Z/C										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SRG	000		A/B	0	D/E	*LS *RS D/E	000 001 010	†CLE D/E	†SL* D/E	*LS *RS R*L R*R *LR ER* EL* *LF	000 001 010 011 100 101 110 111 000			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ASG	000		A/B	1	CL* CM* CC*	01 10 11	CLE CME CCE	01 10 11	SEZ SS* SL*	IN* SZ*	RSS			

Memory Address





## EXTENDED INSTRUCTION GROUP CODES IN BINARY (CONTINUED)

MEMORY EXPANSION	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DJP/DJS	1	0	0	0	1	0	1	1	1	1	0	1	1			

DJP = 0 1 0  
DJS = 0 1 1

SYB:USB:PAB PBB:SSM:JRS	1	0	0	0	1	0	1	1	1	1	0	0	1			
----------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--

SYB = 0 0 0  
USB = 0 0 1  
PAB = 0 1 0  
PBB = 0 1 1  
SSM = 1 0 0  
JRS = 1 0 1

XMA:XLA:XSA XCA:LFA	1	0	0	0	0	0	1	1	1	1	0	1	0			
------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--

XMA = 0 1 0  
XLA = 1 0 0  
XSA = 1 0 1  
XCA = 1 1 0  
LFA = 1 1 1

MBI:MBF:MBW MWI:MWF:MWW	1	0	0	0	1	0	1	1	1	1	0	0	0	0		
----------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--

MBI = 0 1 0  
MBF = 0 1 1  
MBW = 1 0 0  
MWI = 1 0 1  
MWF = 1 1 0  
MWW = 1 1 1

SYA:USA: PAA:PBA	1	0	0	0	0	0	1	1	1	1	0	0	0	1		
---------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--

SYA = 0 0 0  
USA = 0 0 1  
PAA = 0 1 0  
PBA = 0 1 1

**TABLES**

**EXTENDED INSTRUCTION GROUP CODES IN  
BINARY (CONTINUED)**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XMM:XMS/ XMB:XLB/ XSB:XCB/LFB	1	0	0	0	1	0	1	1	1	1	0	1	0			

XMM = 0 0 0

XMS = 0 0 1

XMB = 0 1 0

XLB = 1 0 0

XSB = 1 0 1

XCB = 1 1 0

LFB = 1 1 1

RSA/RVA	1	0	0	0	0	0	1	1	1	1	0	1	1			
---------	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--

RSA = 0 0 0

RVA = 0 0 1

RSB/RVB/SJP/ SJS/UJP/UJS	1	0	0	0	1	0	1	1	1	1	0	1	1			
-----------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--

RSB = 0 0 0

RVB = 0 0 1

SJP = 1 0 0

SJS = 1 0 1

UJP = 1 1 0

UJS = 1 1 1

## SYSTEM COMMUNICATIONS AREA LOCATIONS

Octal Location	Contents	Description
<b>SYSTEM TABLE DEFINITION</b>		
01645	XIDEX	Address of current program's ID extension
01646	XMATA	Address of current program's MAT entry
01647	XI	Address of index register save area
01650	EQTA	FWA of Equipment Table
01651	EQT#	Number of EQT entries
01652	DRT	FWA of Device Reference Table, word 1
01653	LUMAX	Number of logical units in DRT
01654	INTBA	FWA of Interrupt Table
01655	INTLG	Number of Interrupt Table Entries
01656	TAT	FWA of Track Assignment Table
01657	KEYWD	FWA of keyword block
<b>I/O MODULE/DRIVER COMMUNICATION</b>		
01660	EQT1	Addresses of first 11 words of current EQT entry (see 01771 for last four words)
01661	EQT2	
01662	EQT3	
01663	EQT4	
01664	EAT5	
01665	EAT6	
01666	EQT7	
01667	EQT8	
01670	EQT9	
01671	EQT10	
01672	EQT11	
01673	CHAN	Current DCPC channel number
01674	TBG	I/O address of time-base card
01675	SYSTY	EQT entry address of system TTY



**TABLES**

**SYSTEM COMMUNICATIONS AREA LOCATIONS  
(CONTINUED)**

Octal Location	Contents	Description
<b>SYSTEM REQUEST PROCESSOR/EXEC COMMUNICATION</b>		
01676	RQCNT	Number of request parameters -1
01677	RQRTN	Return point address
01700	RQP1	Addresses of request parameters (set for a maximum of nine parameters)
01701	RQP2	
01702	RQP3	
01703	RQP4	
01704	RQP5	
01705	RQP6	
01706	RQP7	
01707	RQP8	
01710	RQP9	
<b>UTILITY PARAMETERS</b>		
01755	TATLG	Negative length of track assignment table
01756	TATSD	Number of tracks on system disc
01757	SECT2	Number of sectors/track on LU2 (system)
01760	SECT3	Number of sectors/track on LU3 (aux.)
01761	DSCLB	Disc address of library entry points
01762	DSCLN	Number of user available library entry points
01763	DSCUT	Disc address of relocatable disc resident library
01764	SYSLN	Number of system library entry points
01765	LGOTK	LGO: LU#, starting track, number of tracks (same format as ID segment word 28)
01766	LGOC	Current LGO track/sector address (same format as ID segment word 26)

## SYSTEM COMMUNICATIONS AREA LOCATIONS (CONTINUED)

Octal Location	Contents	Description
<b>UTILITY PARAMETERS, cont'd.</b>		
01767	SFCUN	LS: LU# and disc address (same format as ID segment word 26)
01770	MPTFL	Memory protect ON/OFF flag (0/1)
01771	EQT12	Address of last four words of current EQT
01772	EQT13	
01773	EQT14	
01774	EQT15	
01775D	FENCE	
01776	VMASWP	VMA swap flag
01777	BGLWA	LWA memory background partition
D letter indicates the contents of the location are set dynamically by the dispatcher.		
<b>SYSTEM LISTS ADDRESSES</b>		
01711	SKEDD	Schedule list
01712	PVCN	Privileged nest counter
01713	SUSP2	Wait Suspend list
01714	SUSP3	Available Memory list
01715	SUSP4	Disc Allocation list
01716	SUSP5	Operator Suspend list
<b>PROGRAM ID SEGMENT DEFINITION</b>		
01717	XEQT	ID segment address of current program
01720	XLINK	Linkage
01721	XTEMP	Temporary (five words)
01726	XPRIO	Priority word
01727	XPENT	Primary entry point
01730	XSUSP	Point of suspension
01731	XA	A-register at suspension
01732	XB	B-register at suspension
01733	XEO	E and overflow register suspension

**TABLES**

**SYSTEM COMMUNICATIONS AREA LOCATIONS  
(CONTINUED)**

Octal Location	Contents	Description
<b>SYSTEM MODULE COMMUNICATION FLAGS</b>		
01734	OPATN	Operator/keyboard attention flag
01735	OPFLG	Operator communication flag
01736	SWAP	RT disc resident swapping flag
01737	DUMMY	I/O address of dummy interface flag
01740	IDSDA	Disc address of first ID segment
01741	IDSDP	Position within disk sector
<b>MEMORY ALLOCATION BASES DEFINITION</b>		
01742	BPA1	FWA user base page link area
01743	BPA2	LWA user base page link area
01744	BPA3	FWA user base page link
01745	LBORG	FWA of resident library area
01746	RTORG	FWA of real-time COMMON
01747	RTCOM	Length of real-time COMMON
01750 D	RTDRA	FWA of real-time partition
01751 D	AVMEM	LWA+1 of real-time partition
01752	BGORG	FWA of background COMMON
01753	BGCOM	Length of background COMMON
01754 D	BGDRA	FWA of background partition

## DEVICE REFERENCE TABLE (DRT)

		SUBCHANNEL NO							EOT ENTRY NUMBER							WORD 1	
		15	14	13	12	11	10	9	8	7	6	5	4	3	2		1
F		DOWNED I/O REQUEST LIST POINTER															WORD 2
LU LOCK OF ODD LU#s								LU LOCK OF EVEN LU#s								WORD 3	

WHERE

F (UP/DOWN FLAG) = 0 IF DEVICE IS UP  
= 1 IF DEVICE IS DOWNLU LOCK = 0 IF NO LOCK ON LU  
= RESOURCE NUMBER BEING USED FOR LOCK

#100-5

## EQUIPMENT TABLE (EQT)

WORD	CONTENTS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O REQUEST LIST POINTER <C>														
2	R	DRIVER INITIATION SECTION ADDRESS <A>														
3	R	DRIVER CONTINUATION/COMPLETION SECTION ADDRESS <A>														
4	D	B	P	S	T	LOWER 5 BITS OF SUBCHANNEL <C>					I/O SELECT CODE # <A>					
5	AV <F>		EQUIPMENT TYPE CODE <A>						STATUS <E>							
6	CONWD (CURRENT I/O REQUEST WORD) <C>													MSB		
7	REQUEST BUFFER ADDRESS <C>															
8	REQUEST BUFFER LENGTH <C>															
9	TEMPORARY STORAGE <D> OR OPTIONAL PARAMETER <C>															
10	TEMPORARY STORAGE <D> OR OPTIONAL PARAMETER <C>															
11	TEMPORARY STORAGE FOR DRIVER <D>															
12	TEMPORARY STORAGE FOR DRIVER <D>						OR	EQT EXTENSION SIZE, ANY <A>								
13	TEMPORARY STORAGE FOR DRIVER <D>						OR	EQT EXTENSION STARTING ADDRESS, IF ANY <A>								
14	DEVICE TIME-OUT RESET VALUE <B>															
15	DEVICE TIME-OUT CLOCK <C>															

#100-6

## TABLES

### LEGEND FOR EQT TABLE

- R = reserved for system use.
- I/O Request List Pointer = points to list of requests queued up on this EQT entry.
- D = 1 if DCPC required
- B = 1 if automatic output buffering used.
- P = 1 if driver is to process power fail.
- S = 1 if driver is to process time-out.
- T = 1 if device timed out (system sets to zero before each I/O request).
- Subchannel = last subchannel addressed. (lower 5 bits)  
MSB = most significant bit of the subchannel (bit 6).
- I/O Select Code = I/O select code for the I/O controller (lower number if a multi-board interface).
- AV = I/O controller availability indicator:  
0 = available for use.  
1 = disabled (down).  
2 = busy (currently in operation).  
3 = waiting for an available DCPC channel.
- Equipment Type Code = type of device on this controller. When this octal number is linked with "DVy," it identifies the device's software driver routine. Some standard driver numbers are:  
00 to 07 = paper tape devices or consoles  
00 = teleprinter or keyboard control device  
01 = photoreader  
02 = paper tape punch  
05 = 264x-series terminals  
07 = multi-point devices

## LEGEND FOR EQT TABLE (CONTINUED)

	10 to 17 = unit record devices
	10 = plotter
	11 = card reader
	12 = line printer
	15 = mark sense card reader
	20 to 37 = magnetic tape/mass storage devices
	23 = 9-track magnetic tape (800/1600 BPI)
	31 = 7900 moving head disc
	32 = 7905/06/20/25 moving head disc
	33 = 7908/11/12/35 moving head disc drive, cartridge tape drive or 9895 flexible disc drive.
	36 = writable control store
	37 = HPIB
	40 to 77 = instruments
STATUS	= actual physical status or simulated status at the end of each operation (see Device Status Table).
CONWD	= combination of user control word and user request code word in the I/O EXEC call (see EQT wd. 6).
Letters in brackets (<>) indicate the nature of each data item as follows:	
	<A> = fixed at generation or reconfiguration time; never changes
	<B> = fixed at generation or reconfiguration time; can be changed on-line
	<C> = set up or modified at each I/O initialization
	<D> = available as temporary storage by driver
	<E> = can be set driver
	<F> = maintained by system

**TABLES**

**DEVICE STATUS TABLE A**

Device/Status	7	6	5	4	3	2	1	0
Teleprinter(s) Photoreader(s) Punch(s) DVR00	X	—	End of I/O Tape	—	—	STL	TEN	—
262x 263x 264x Terminal Cartridge Tape Unit DVR05, DVA05	BF	—	CD	—	—	—	TEN	—
	EOF	TLP	EOT	RE	LCA	CWP	EOD	CNI/ DB
2892A Card Reader DVR11	HE/ SOR	SF	HE/ SF	PF	TE/ PF	OL	ICC/ HF	RNR
2607 Line Printer 2610 Line Printer 2613/17/18 Line Printer 2631 Line Printer DVA12	—	TOF	—	ID	PSE	OL	—	—
	—	TOF	—	ID	SSE	PO	—	—
	—	TOF	—	ID	ON	NR	V9	V12
	—	TOF	—	BR	ON	PO	—	—
2608A Line Printer DVB12	PW	TOF	S8	VI	ON	NR	V9	V12
2607A Line Printer DVR12	TOF	DM	ON	RX	—	—	APE	—

DEVICE STATUS TABLE A (CONTINUED)

Device/Status	7	6	5	4	3	2	1	0
7261A Card Reader DVR15	EOF	—	HF/ SF	PF	—	—	DE	RNR
7970 Mag Tape DVR23	EOF	ST	EOT	TE	I/O R	NW	PE/ TE	OL
7900 Moving Head Disc DVR31	—	NR	EOT	AE	FC	SC	DE	EE
79XX Disc Drives DVR32	PS	FS	HF	FC	SC	NR	DB	EE
79XXH, 9895 Disc Drives DVA32	PS	FS	HF	FC	SC	NR	DB	EE
CS80 Disc Drives DVM33 See Status Table B DVR33	WP	RER/ UD	EOF/ EOV	UN	FA	NR	CHE	SE
59310B HPIB DVR37	—	EF	II/O	NOA	SRQA	IFC	TO	—



## TABLES

### DEVICE STATUS TABLE B

DVR33

127323A, 12733A Disc Drives

Bits 0-7	Meaning
00000000	No Error
00000011	No Drive Power
00000101	Door Open
00000111	No Disc
00001011	Record Not Found
00001101	Track Not Found
00001111	Data Checkword Error
00010001	Data Overrun
00010011	Read "Tight Margin" Error
00011111	Transfer Incomplete
00100001	Data Block too long
00100000*	End of Track (Access track > 66)
01000000*	Disc Change
10000000*	Disc Write Protected

DVA47

Serial Link Drive

Bits 0-7	Meaning
00000001	Time out occurred
00000010	Hardware Failure
00000011	Hardware Failure on Controller
00000100	Bad System Configuration
00000101	Illegal Request

## DEVICE STATUS TABLE KEY

AE	= Address Error
AF	= Abort Flag (NR (Bit = 7 = 0) has occurred during since last data transfer)
APE	= Auto Page Eject
BF	= Buffer Flushed
BR	= Buffer Ready
BT	= Broken Tape
CD	= Control-D Entered
CE	= Compare Error
CHE	= Channel Error
CNI	= Cartridge Not Inserted
CWP	= Cartridge Write Protected
DB	= Device Busy
DE	= Data Error
DF	= Drive or Controller Fault
DM	= Demand (1 = idle)
DR	= Disc Ready
EE	= Error Exists
EF	= EQT Extension Area Full
EOD	= End of Data
EOF	= End of File
EOT	= End of Track
EOV	= End of Volume
FA	= Drive or Controller Fault
FC	= Flagged Track
FS	= Driver Format Switch is Set
HE	= Hopper Empty
HF	= Hardware Fault
ICC	= Illegal Card Code
ID	= Idle
IFC	= IFC Detected
II/O	= Illegal I/O Request
I/OR	= I/O Reject
LCA	= Last Command Aborted
LCF	= Last Character Flag
NE	= No Error
NOA	= Non-existent alarm program
NR	= Not Ready
NW	= No write (ring missing or rewinding)
OL	= Off Line
ON	= On Line
PD	= Pen Down
PE	= Parity Error

**DEVICE STATUS TABLE KEY  
(CONTINUED)**

PF	= Pick Fail
PW	= Power Fail
PO	= Paper Out
PS	= Protect Switch Set
PSE	= Print Switch Enabled
RD	= Release Drive
RE	= Read Error
RER	= Recoverable Error
RNR	= Reader Not Ready
RX	= Ready (0= Power On)
SAC	= Sector Address Coincidence
SC	= Seek Check
SE	= Severe Error
SF	= Stacker Full
SOR	= EOF Switch on during Read
SSE	= Start Switch enabled
ST	= Start of Tape
STL	= Stall required in program
S8	= Set is 8 LPI
TE	= Timing Error
TEN	= Terminal Enabled
TLP	= Tape at Load Pt
TO	= Device Time Out
TOF	= Top of Form
UD	= Unrecoverable Data
UN	= Uninitialized Media
VI	= VFC Initialized
V9	= VFU Chan 9 detected
V12	= VFU Chan 12 detected
WE	= Currently addressed track is write enabled
WP	= Write Protect
X	= Driver internal use

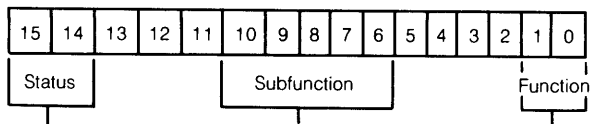
## Device File DCB

Word 0	0
1	0
2	File Type (0)
3	XLUEX LU Word
4	XLUEX Function Word
5	Spacing Flags
6	EOF Function Code
7	Read/Write Flags
8	0
9	Program ID Segment Address
10	0
11	0
12	0
13	32-Bit Record Number
14	
15	0

8300-61

# TABLES

## EQT WORD 6



- |                    |                          |                   |
|--------------------|--------------------------|-------------------|
| 00 — standard call | 00000 = clear controller | 01 — READ call    |
| 01 — buffered call | (if function = 11 =      | 10 — WRITE call   |
| 10 — system        | CONTROL call)            | 11 — CONTROL call |
| 11 — Class call    |                          |                   |

Other subfunctions are driver specific and may or may not be defined

## ID SEGMENT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word	
List Linkage																0	← XEQOT
TEMP 1																1	
TEMP 2																2	
TEMP 3																3	
TEMP 4																4	
TEMP 5																5	
Priority																6	Memory-Resident Programs
Primary Entry Point																7*	
Point of Suspension																8	
A-Register																9	
B-Register																10	
EO-Registers																11	
Name 1								Name 2								12*	
Name 3								Name 4								13*	
Name 5								TM	ML	TS	SS	Type				14*	
NA	NS	NP	W	A	FS	O	LP	R	D					Status		15	
Time List Linkage																16	
RES				T		Multiple										17	
Low Order 16 Bits of Time																18	
High Order Bits of Time																19	
BA	FW	M	AT	RM	RE	PW	RN	Father ID Segment No.								20	
RP	# pgs. (no BP)					MPFI			DE	Partition No. - 1						21	
Low Main Address																22*	
High Main Address + 1																23*	
Low Base Pg Addr (Non-MLS Prog) or # Sectors on LU 2 or 3 for MLS Prog																24*	
High Base Page Address + 1																25*	
Program: Track #																26*	
LU								Swap: Track #								27	
ID Extension No.								EMA Size								28	
High Address + 1 of Largest Segment or Node																29	
Timeslice word																30	
SEQCNT				SH	DC	CP	DS	Session ID								31	
SCB Pointer																32	
MS	# pages disc resident					# pages memory-resident										33	
MP	# pgs dynamic buffer area					E	DB	# of swap tracks								34	
Start sector address of program								LU # of prog								35	

8100-2A

- TM = temporary load (copy of ID segment is not on the disc)
- ML = memory lock (program may not be swapped)
- TS = the program is transportable
- SS = short segment (indicates a nine-word segment)
- TYPE = specified program type (1-6)
- NA = no abort (instead, pass abort errors to program)
- NS = no suspension on I/O requests (instead, pass control to program)

## TABLES

### ID SEGMENT LEGEND (CONTINUED)

NP	= no parameters allowed on reschedule
W	= wait bit (waiting for program whose ID segment address is in word 1)
A	= abort on next list entry for this program
FS	= file system bit
O	= operator suspend on next schedule attempt
LP	= load in progress; program is being dispatched from disc
R	= resource save (save resources when setting dormant)
D	= dormant bit (set dormant on next schedule attempt)
Status	= current program status
T	= time list entry bit (program is in the time list)
BA	= batch (program is running under batch)
FW	= father is waiting (father scheduled with wait)
M	= Multi-Terminal Monitor bit
AT	= attention bit (operator has requested attention)
RM	= reentrant memory must be moved before dispatching program
RE	= reentrant routine now has control
PW	= program wait (some other program wants to schedule this one)
RN	= Resource Number either owned or locked by this program
RP	= reserved partition (only for programs that request it)
MPFI	= memory protect fence index
DE	= defer EXEC 6 (terminate program) request
LU	= 0 if LU 2, 1 if LU 3
SEQCNT	= sequence counter
SH	= shareable EMA flag (program or progeny uses shareable EMA)
DC	= don't copy flag
CP	= copy flag
DS	= DS program
Session ID	= system LU of terminal where program was loaded
MS	= multi level segmentation flag
MP	= program is using modified maps for I/O
E	= EXEC 4 (track allocation) request was made by program
DB	= Debug bit
*	= words used in short ID segments for program segments

## ID SEGMENT EXTENSION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED															WORD 0
MSEG START PAGE (LOGIC.)				DE	(PHYSICAL) EMA START PAGE										WORD 1
COM	SW		INDEX # INTO \$EMTB												WORD 2
LAST VIRTUAL PAGE #															WORD 3
RESERVED															WORD 4

8100-7

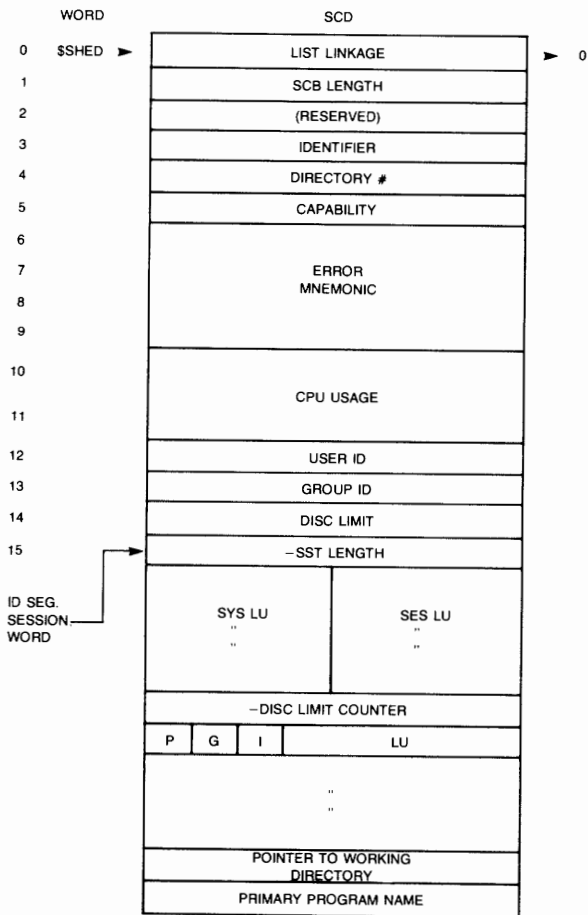
WHERE:

- DE = 0 IF THE EMA SIZE WAS SPECIFIED BY THE USER  
1 IF THE EMA SIZE IS ALLOWED TO DEFAULT TO THE MAXIMUM SIZE AVAILABLE TO THE SYSTEM
- COM = 1 IF THE PROGRAM IS USING SHAREABLE EMA
- SW = 0 PTE TABLE DOES NOT CONTAIN VALID DATA  
SW = 1 PTE TABLE IS STILL INTACT.



# TABLES

## SESSION CONTROL BLOCK (SCB)



8300-222

- P = ADDED SST ENTRY FOR THIS DISC
- G = THIS IS A GROUP CARTRIDGE
- I = THIS DISC CARTRIDGE IS INACTIVE

## SYSTEM DISC LAYOUT

	AVAILABLE DISC SPACE	
	CARTRIDGE LIST	
	LIBRARY ENTRY POINTS LIST *	← DISC PROTECT BOUNDARY
Δ	RELOCATABLE LIBRARY AND UTILITIES	
Δ	BASE PAGE LINKS	} REPEATED FOR ALL BG DISC RESIDENTS AND SEGMENTS
Δ	BACKGROUND DISC RESIDENT	
Δ	BASE PAGE LINKS	} REPEATED FOR ALL RT DISC RESIDENTS AND SEGMENTS
Δ	REAL-TIME DISC RESIDENT	
Δ	MEMORY RESIDENT PROGRAMS	
Δ	MEMORY RESIDENT LIBRARY	
Δ	MEMORY RESIDENT BASE PAGE	
Δ	PARTITION RESIDENT DRIVERS	
Δ	PARTITION RESIDENT O.S. CODE	
	SYSTEM	
	TYPE 13 MODULES \$ MATA, \$ MRMP, \$ MPFT TABLES KEYWORD TABLE, ID SEGMENTS ID EXTENSIONS, \$ IDEX TABLE \$ CLAS, \$ LUSW, \$ RNTB, \$ LUAV TABLES SHAREABLE EMA TABLE \$EMTB	} TABLE AREA II
Δ	SYSTEM DRIVER AREA	
	BACKGROUND COMMON REAL-TIME COMMON SSGA	} COMMON
Δ	PARTITION #1 RESIDENT DRIVERS	
Δ	TYPE 15 MODULES INT DRT \$ DVMP TABLE EQT, EOT EXTENSIONS	} TABLE AREA I
Δ	SYSTEM COMMUNICATION AREA UPPER BASE PAGE LINKS SYSTEM LINKS TRAP CELLS	} SYSTEM BASE PAGE
Δ	BOOT EXTENSION	

Δ SECTOR BOUNDARIES

\* INCLUDES ONE SYSTEM-RESERVED TRACK

8100-8

TABLES

Data Control Block (DCB) Format

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
WORD 0*	SECTOR OFFSET			SECTOR # OF FILE DIRECTORY				LU # OF FILE DIRECTORY OR OF FILE IF ON DISC						FILE DIRECTORY ADDRESS				
1*	TRACK # OF FILE DIRECTORY																	
2	FILE TYPE (MAY BE OVERRIDDEN AT OPEN, UNLESS TYPE 0)																	
3	TRACK ADDRESS OF FILE (TYPE ≥ 1)						OR		LU # OF FILE (TYPE = 0)									
4	SECTOR ADDRESS OF FILE (TYPE ≥ 1)						OR		EXEC FUNCTION CODE (TYPE = 0)									
5	FILE SIZE IN -CHUNKS +SECTORS (TYPE ≥ 1)						OR		SPACING CODE (TYPE = 0)									
6	RECORD LENGTH (TYPE = 2)						OR		END-OF-FILE CODE (TYPE = 0)									
7	WA	PF	NUMBER OF BLOCKS IN DCB BUFFER				FM	EX	RA	OM	IB	EF	WR					
8	NUMBER OF SECTORS PER TRACK (TYPE ≥ 1)																	
9	OPEN-CLOSE INDICATOR																	
10	TRACK # OF CURRENT FILE POSITION (TYPE ≥ 1)						OR		THE VALUE OF THE A-REGISTER AFTER THE LAST EXEC CALL (TYPE = 0)						CURRENT POSITION IN FILE			
11	TRACK # OF CURRENT FILE POSITION (TYPE ≥ 1)						OR		THE VALUE OF THE B-REGISTER AFTER THE LAST EXEC CALL (TYPE = 0)									
12	LOCATION OF NEXT WORD IN FILE (TYPE ≥ 1)																	
13	RECORD # OF CURRENT FILE																	
14	POSITION (DOUBLE WORD INTEGER)																	
15	EXTENT NUMBER (TYPE ≥ 3)																	
16	DCB BUFFER AREA (128 - N)																	

8100-4A

\*FILE DIRECTORY ADDRESS

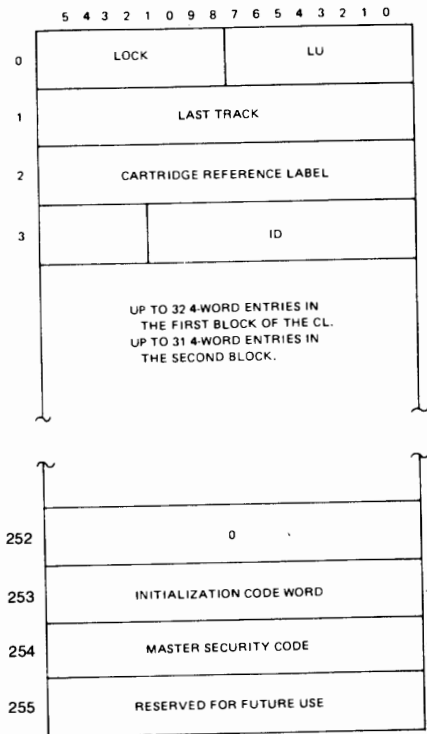
## LEGEND FOR DATA CONTROL BLOCK

WORD	CONTENT
0 File Directory Address:	bits 6-12 = Physical sector number of the file directory. bits 13-15 = Entry offset from the beginning of the block (origin 0).
5 Spacing Code: (type 0 file)	bit 15 = 1 — backspace legal. bit 0 = 1 — forward space legal.
6 End-of-File Code: (type 0 file)	01 lu = EOF on Magnetic Tape. 10 lu = EOF on Paper Tape. 11 lu = EOF on Line Printer.
7 Status Information	
(WA) Write Allowed:	bit 15 = 1 — write to file allowed. = 0 — write to file now allowed.
(PF) Partially Full:	bit 14 = 1 — DCB is only partially full. = 0 — DCB is full.
DCB Buffer:	bits 13-7 = Number of blocks in the DCB buffer.
(FM) File Modify:	bit 6 = 1 — File has been modified. = 0 — File has not been modified.
(EX) Extendable:	bit 5 = 1 — File is not extendable. = 0 — File is extendable.
(RA) Read Allowed:	bit 4 = 1 — Read from file allowed. = 0 — Read from file not allowed.

## LEGEND FOR DATA CONTROL BLOCK (CONTINUED)

WORD	CONTENT
(OM) Open Mode:	bit 3 = 1 — update open 0 — standard open
(IB) In Buffer Flag:	bit 2 = 1 — data in DCB buffer = 0 — data not in DCB buffer
(EF) EOF Read Flag:	bit 1 = 1 — EOF has been read = 0 — EOF has not been read
(WR) To Be Written:	bit 0 = 1 — data in DCB buffer to be written = 0 — data in DCB buffer not to be written
9	Open/Close Indicator: if open, contains ID segment location of program performing open. If closed, set to zero.

# CARTRIDGE DIRECTORY FORMAT



NOTE: IF THE CARTRIDGE REFERENCE LABEL IS EQUAL TO ZERO, THIS IS A HIERARCHICAL FILE VOLUME.

SUM OF CONTENTS OF BASE PAGE WORDS 1650 THRU 1657 AND 1742 THRU 1747 AND 1755 THRU 1764.  
SET WHEN SYSTEM CARTRIDGE IS INITIALIZED.

LOCK = 0 IF NOT LOCKED; ELSE IS KEYWORD TABLE OFFSET OF ID SEGMENT ADDRESS OF LOCKING PROGRAM

LOCKED DISCS ARE AVAILABLE ONLY TO THE LOCKER.

ID IDENTIFIES TO WHOM THE CARTRIDGE IS MOUNTED.

- ID = 0000 → NON-SESSION
- ID = 7777 → SYSTEM CARTRIDGE
- 0 < ID < 7777 → SESSION MONITOR GROUP OR PRIVATE CARTRIDGE

NOTE: WORDS 124, 125, 126, AND 127 ARE UNIQUE ONLY IN THE SECOND BLOCK OF THE CL. THE FIRST BLOCK WILL HOLD 32 ENTRIES IN WORDS 0 THROUGH 127.

# TABLES

## FILE DIRECTORY

BIT WORD	Content															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Six-character cartridge label															
1																
2																
3	Cartridge Reference Number															
4	First Available Track for FMP															
5	CPU	F	Next Available Sector													
6	Number of Sectors per Track															
7	Lowest Directory Track (last file track + 1)															
8	Number of Tracks in Directory (negative value)															
9	Next Available FMP Track															
10	First Bad Track															
11	Second Bad Track															
12	Third Bad Track															
13	Fourth Bad Track															
14	Fifth Bad Track															
15	Sixth Bad Track															

Bit 15 set to distinguish cartridge entry from file entry.

#100-13

## DISC DIRECTORY FILE ENTRY

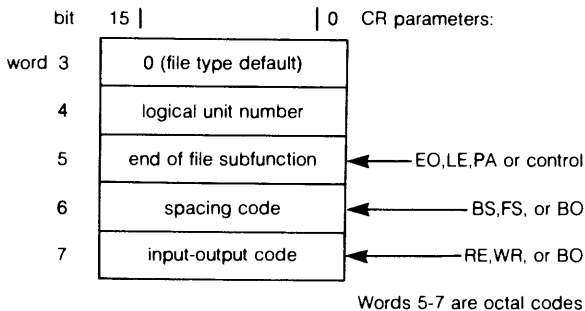
WORD	BIT	15	8	7	0
0	6-CHARACTER FILE NAME				
1					
2					
3	FILE TYPE (1 THRU 32767)				
4	STARTING TRACK				
5	EXTENT #		STARTING SECTOR		
6	FILE SIZE IN + SECTOR OR - CHUNKS				
7	RECORD LENGTH (TYPE 2 ONLY)				
8	SECURITY CODE				
9					
10					
11	OPEN FLAGS				
12	15 = 1 FOR EXCLUSIVE OPEN				
13	14-12 = RESERVED				
14	11-8 = SEQUENCE COUNTER				
15	7-0 = KEYWORD OFFSET OF OPENING PROGRAM'S ID SEGMENT				

WORD 0 = 0 IF THE LAST ENTRY IN  
DIRECTORY; = -1 IF FILE IS PURGED



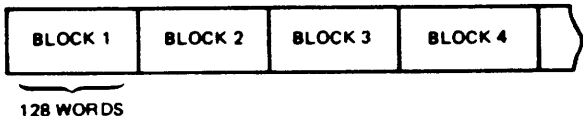
## DISC DIRECTORY TYPE 0 FILE ENTRY

The entries for non-disc (type 0) files differ from those for disc files in words 3 through 7:



## DISC FILE RECORD FORMATS

Fixed Length Formats (Types 1 and 2)



1st 127 WORDS OF LAST BLOCK

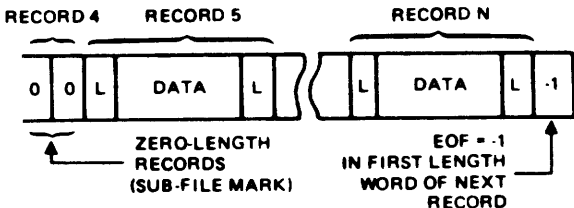
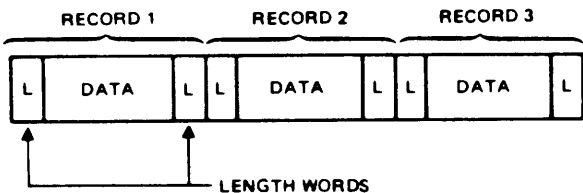


EOF FOLLOWS LAST WORD IN LAST BLOCK

Type 1 Record length = Block length = 128 words

Type 2 Record length is user defined; may cross block boundaries but not past EOF

Variable Length Formats (Types 3 and Above)



# TABLES

## TYPE 6 FILE FORMAT

Files created by the SP command as memory-image program files are always accessed as type 1 files (fixed length, 128-words per record).

WORD	CONTENT	
0	- 1	EOF UNLESS FORCED TO TYPE 1
1-5	NOT USED	
6	PRIORITY	
7	PRIMARY ENTRY POINT	
8-11	NOT USED	
12-13	ORIGINAL PROGRAM NAME	
14	PROGRAM TYPE	
15-16	NOT USED	
17-19	TIME PARAMETERS	
20	SUBSTATUS 1 - WORD 20 OF ID SEGMENT	
21	SUBSTATUS 2 - WORD 21 OF ID SEGMENT	
22	LOW MAIN ADDRESS	
23	HIGH MAIN ADDRESS + 1	
24	LOW BASE-PAGE ADDRESS	
25	HIGH BASE-PAGE ADDRESS + 1	
26	PROGRAM TRACK	
27	SWAP TRACK	
28	ID EXT. #/EMA SIZE	
29	HIGH ADDRESS + 1 OF LARGEST SEGMENT	
30	NOT USED	
31	OPEN FLAG WORD	
32	NOT USED	
33	MLS WORD 1	
34	MLS WORD 2	
35	SECTOR/LU OF PROGRAM	
36	CHECKSUM OF WORDS 0 - 32	SUM OF CONTENTS OF WORDS 1650 THRU 1657 AND WORDS 1742 THRU 1747 AND 1755 THRU 1764 IN BASE PAGE
37	SETUP CODE WORD	
38	ID EXTENSION - WORD 0	
39	ID EXTENSION - WORD 1	
40	ID EXTENSION - WORD 2	
41	ID EXTENSION - WORD 3	
42	ID EXTENSION - WORD 4	
43-45	SHARED EMA NAME	
46	OWNER ID	IF SIGN BIT SET, PROGRAM FILE PROTECTED TO THIS USER ID
47	OWNER'S GROUP ID	IF SIGN BIT SET, PROGRAM FILE PROTECTED TO THIS GROUP ID
48	CAPABILITY LEVEL REQUIRED	MINIMUM CAPABILITY REQUIRED TO RU OR RP THIS PROGRAM.
49-112	NOT USED	
113-123	TIME TYPE-6 FILE CREATED	
124-127	NOT USED	

WORDS 0-35 AND 38-42 CONTAIN PROGRAM'S ID-SEGMENT INFORMATION

REMAINDER OF FILE IS AN EXACT COPY OF THE PROGRAM BEING SAVED.

8200-162

## Disc Volume Header Format

Word: 1 8

Uppercase ASCII "VOLUME HEADER" (no parity)
---

Word: 9 10 11 12 16

Bit Map	Blk #	Res	Unused
---------	-------	-----	--------

Word: 17 24

Unused
--------

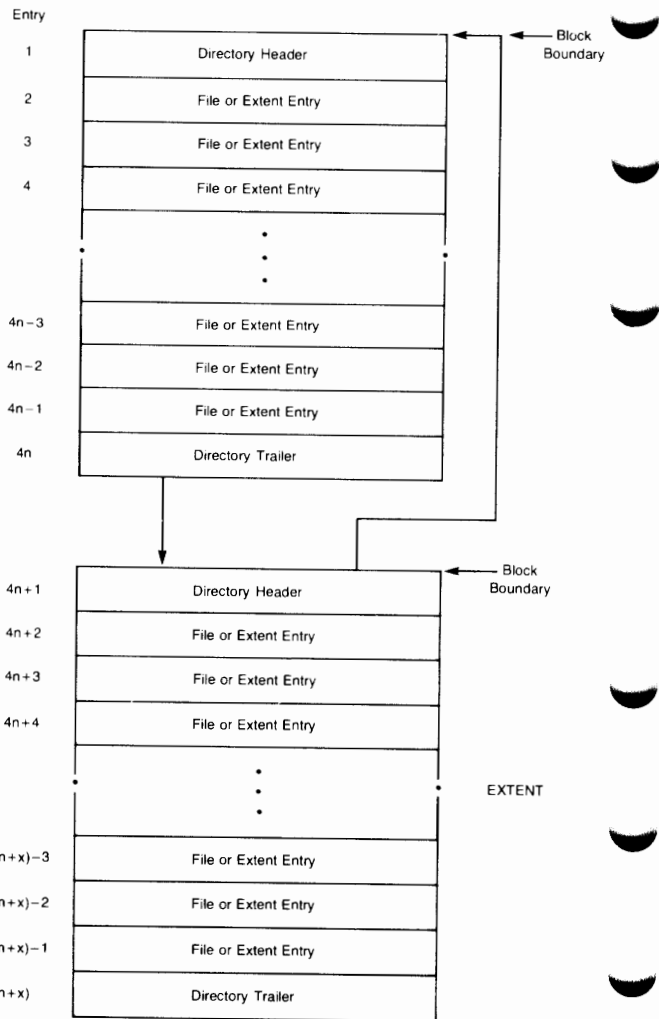
Word: 25 32

Unused
--------

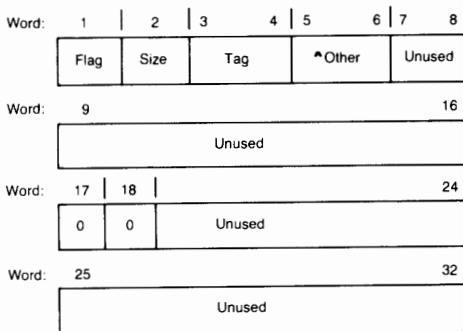
8300-72

**TABLES**

**Directory Structure**

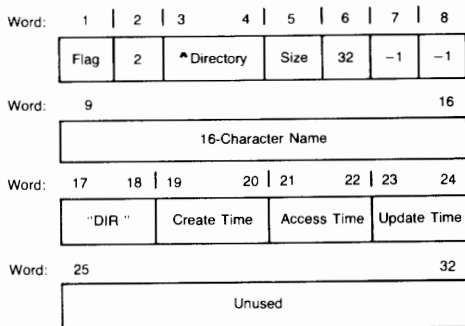


## Root Directory Header/Trailer



8300-70

## Root Directory Entry



8300-71

## TABLES

### Directory Header/Trailer Format

Word:	1	2	3	4	5	6	7	8
	Flag	Size	Tag	^Other		^Parent		

Word:	9							16
	16-Character Name							

Word:	17	18						24
	Owner	Not Currently Used						

Word:	25							32
	Not Currently Used							

8300-69

### File Entry

Word:	1	2	3	4	5	6	7	8
	Flag	Type	^File		Size	Reclen	^Extent	

Word:	9							16
	16-Character Name							

Word:	17	18	19	20	21	22	23	24
	Type Ext.	Create Time		Access Time		Update Time		

Word:	25	26	27	28	29	30	31	32
	nblocks	^eof		nrecords		openflag		

8300-76

## Subdirectory Entry

Word:	1	2	3	4	5	6	7	8	
	Flag	2	^Directory	Size	32	-1	-1		
Word:	9	16-Character Name						16	
Word:	17	18	19	Time Stamps					24
	"DIR "								
Word:	25	Unused						32	

8300-63

## Extent Entry

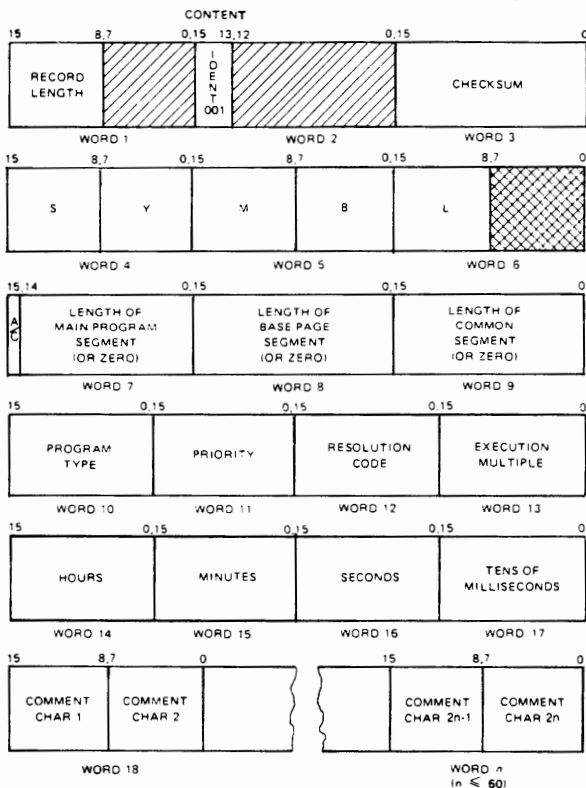
Word:	1	2	3	4	5	6	7		
	Flag	^Ext1	Size1	^Ext2	Size2				
Word:	8	9	10	11	12	13	14	15	16
	^Ext3	Size3	^Ext4	Size4	^Ext5	Size5			
Word:	17	18	19	20	21	22	23	24	
	^Ext6	Size6	^Ext7	Size7	^Ext8				
Word:	25	26	27	28	29	30	31	32	
	Size8	^Ext9	Size9	^Previous	^Next				

8300-62



# TABLES

## NAM RECORD



HATCH-MARKED AREAS SHOULD BE ZERO-FILLED WHEN THE RECORDS ARE GENERATED

CROSS-HATCH-MARKED AREAS SHOULD BE SPACE-FILLED WHEN THE RECORDS ARE GENERATED

### EXPLANATION

RECORD LENGTH = 9-60 WORDS

IDENT = 001

CHECKSUM ARITHMETIC  
TOTAL OF ALL WORDS  
IN RECORD EXCLUDING  
WORDS 1 AND 3

SYMBL FIVE CHARACTER

NAME OF PROGRAM

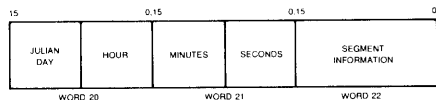
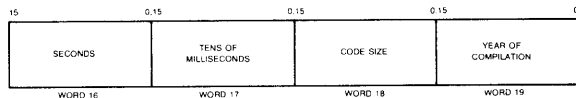
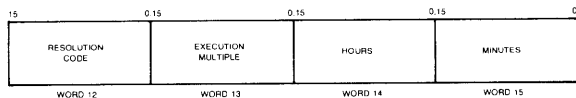
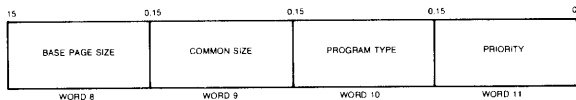
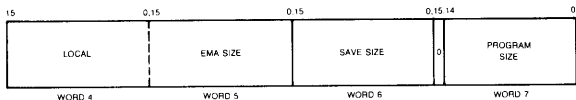
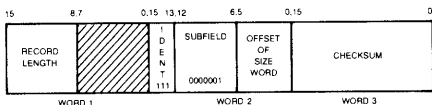
ACC BINARY TAPE PRECESSION

= 0 IF ASSEMBLER PRODUCED  
OR LENGTH IS EXACT

= 1 IF COMPILER PRODUCED  
AND LENGTH IS UNKNOWN

# XNAM RECORD

CONTENT



8100-18

**TABLES**

**XNAM RECORD (CONTINUED)**

CONTENT

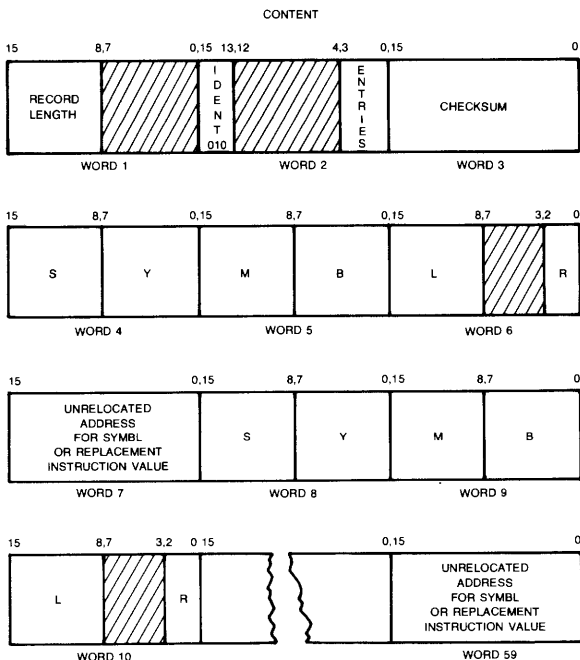
15	0,15	0,15	8,7	0
DATE AND/OR	REVISION CODE	# WORDS OF COMMENT	# WORDS OF SYMBOL	
WORD 23	WORD 24	WORD 25		

SYMBOL	VARIABLE LENGTH
WORD 26	

COMMENT	VARIABLE LENGTH
---------	--------------------

8100-19

# ENT RECORD



## EXPLANATION

RECORD LENGTH = 7-59 WORDS

IDENT = 010

ENTRIES: 1 TO 14 ENTRIES  
PER PROGRAM; EACH ENTRY  
IS FOUR WORDS LONG.

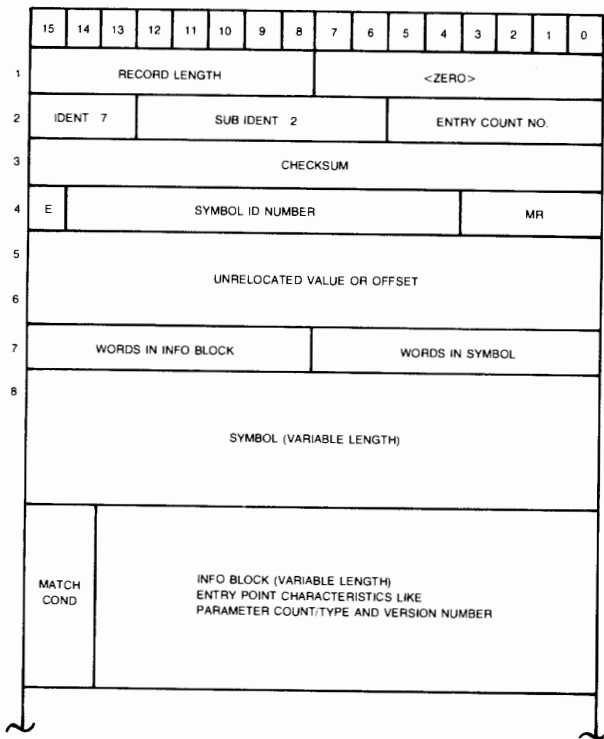
SYMBL: 5 CHARACTER ENTRY  
POINT SYMBOL

R: RELOCATION INDICATOR  
= 0 IF PROGRAM RELOCATABLE  
= 1 IF BASE PAGE RELOCATABLE  
= 2 IF COMMON RELOCATABLE  
= 3 IF ABSOLUTE  
= 4 MICROCODE REPLACEMENT

WORDS 4 THROUGH 7 ARE  
REPEATED FOR EACH  
ENTRY POINT SYMBOL.

**TABLES**

**EXTENDED ENT RECORD (XENT)**



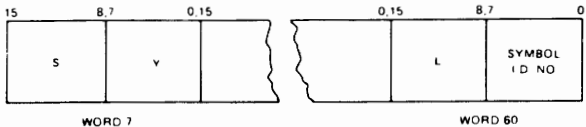
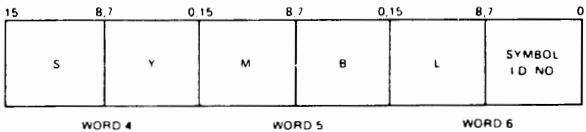
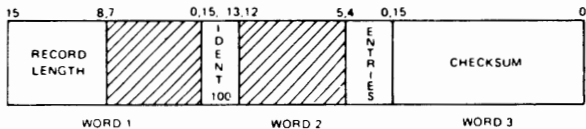
\*THE E BIT DETERMINES THE TYPE OF THE RESULT (ADDR/EMA)

\*THE MAXIMUM ENTRY COUNT NO. IS 25 IN XENT.

8100-20

# EXT RECORD

## CONTENT



## EXPLANATION

RECORD LENGTH = 660 WORDS

IDENT = 100

ENTRIES 1 TO 19 PER RECORD. EACH ENTRY IS THREE WORDS LONG

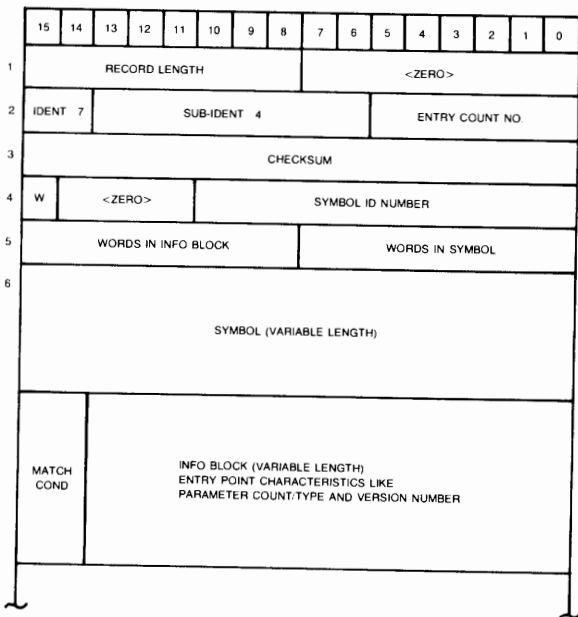
SYMBL 5 CHARACTER EXTERNAL SYMBOL

SYMBOL ID NO NUMBER ASSIGNED TO SYMBL FOR USE IN LOCATING REFERENCE IN BODY OF PROGRAM.

WORDS 4 THROUGH 6 REPEATED FOR EACH EXTERNAL SYMBOL (MAXIMUM OF 19 PER RECORD)

**TABLES**

**EXTENDED EXT RECORD (XEXT)**

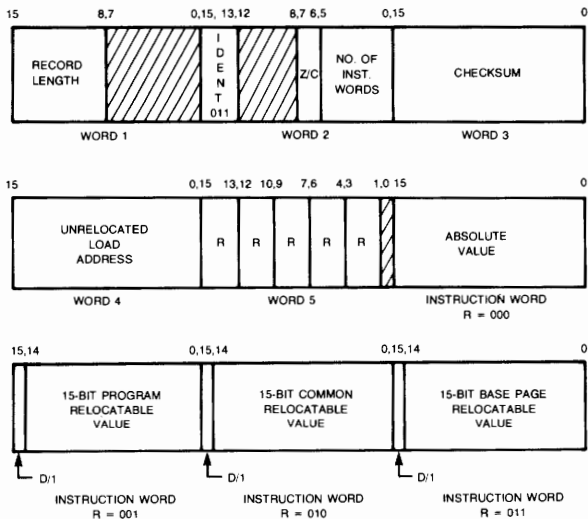


\*W = 1 FOR A WEAK EXTERNAL  
\*THE MAXIMUM ENTRY COUNT IS 41 IN XEXT

8100-21

## DBL RECORD

## CONTENT



## EXPLANATION

RECORD LENGTH = 6-60 WORDS

IDENT = 011

Z/C: RELOCATION OF LOAD ADDRESS

= 0 FOR BASE PAGE

= 1 FOR PROGRAM

= 2 FOR ABSOLUTE

= 3 FOR COMMON

NO. OF INST. WORDS: 1 TO 45  
LOADABLE INSTRUCTION  
WORDS PER RECORDRELOCATABLE LOAD ADDRESS:  
STARTING ADDRESS FOR  
LOADING THE INSTRUCTIONS  
WHICH FOLLOW:

R's: RELOCATION INDICATORS:

000 = ABSOLUTE

001 = 15-BIT PROGRAM  
RELOCATABLE010 = 15-BIT BASE PAGE  
RELOCATABLE011 = 15-BIT COMMON  
RELOCATABLE

100 = EXTERNAL REFERENCE

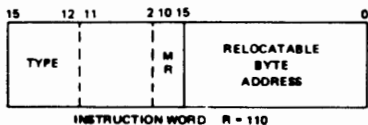
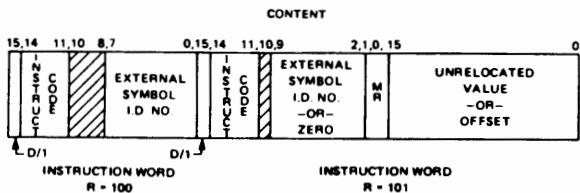
101 = MEMORY REFERENCE

R<sub>1</sub> IS RELOCATION INDICATOR  
FOR INSTRUCTION WORD<sub>1</sub>; R<sub>2</sub>  
FOR INSTRUCTION WORD<sub>2</sub>; ETC.



TABLES

DBL RECORD (CONTINUED)



EXPLANATION

D/I: INDIRECT ADDRESSING

- 0 - DIRECT
- 1 - INDIRECT

MEMORY REFERENCE INSTRUCTIONS USE TWO WORDS, WITHIN THE TWO-WORD GROUP, "MR" INDICATES RELOCATABILITY OF OPERAND SPECIFIED IN SECOND WORDS:

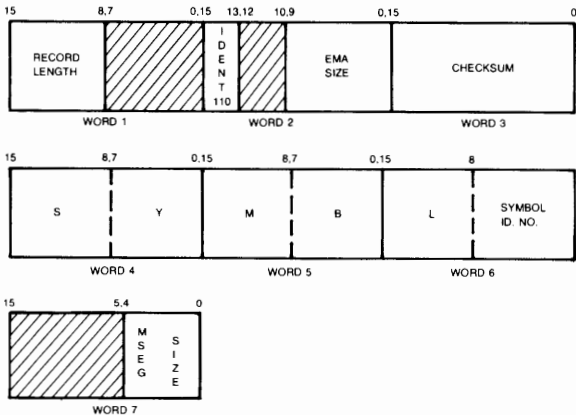
- 00 - PROGRAM RELOCATABLE
- 01 - BASE PAGE RELOCATABLE
- 10 - COMMON RELOCATABLE
- 11 - ABSOLUTE

EXTENDED DBL RECORD (XDBL)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	RECORD LENGTH								<ZERO>								
2	IDENT 7		SUB-IDENT 3					<ZERO>									
3	CHECKSUM																
4	0	SYMBOL ID NUMBER										MR					
5	UNRELOCATED LOAD ADDRESS																
6	WORDS TO RELOCATE								NO. OF "R" VALUES								
7	1ST R				2ND R				3RD R				4TH R				
	ABSOLUTE VALUE																R=0
	PROGRAM RELOCATABLE VALUE																R=1
	BASE PAGE RELOCATABLE VALUE																R=2
	COMMON RELOCATABLE VALUE																R=3
	5TH R				6TH R				7TH R				8TH R				
	1	OPCODE				SYMBOL ID NUMBER											R=4
	PURE CODE RELOCATABLE VALUE																R=5
	0	SYMBOL ID NUMBER										MR				R=6	
	UNRELOCATED VALUE OR OFFSET																
	SAVE DATA RELOCATABLE VALUE																R=7
	9TH R				10TH R				11TH R				12TH R				
	1	OPCODE				<ZERO>						MR				R=8	
	UNRELOCATED VALUE (AFTER OFFSET)																
	SYMBOL ID NUMBER										MR						R=9
	UNRELOCATED BYTE VALUE OR OFFSET														LR		
	1	OPCODE				SYMBOL ID NUMBER											R=10
	<ZERO>										MR						
	UNRELOCATED OFFSET VALUE																

TABLES

EMA RECORD



EXPLANATION

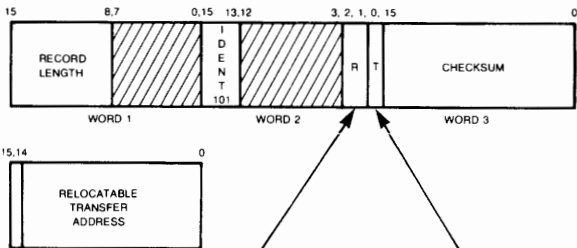
RECORD LENGTH = 7 WORD  
IDENT = 110

SYMBOL ID. NO.: NUMBER  
ASSIGNED TO SYMBL FOR  
USE IN LOCATING REFER-  
ENCE IN BODY OF PROGRAM.

8100-9

END RECORD

CONTENT



R: RELOCATION INDICATOR  
FOR TRANSFER ADDRESS

- = 0 IF PROGRAM RELOCATABLE
- = 1 IF BASE PAGE RELOCATABLE
- = 2 IF COMMON RELOCATABLE
- = 3 IF ABSOLUTE

T: TRANSFER ADDRESS  
INDICATOR

- = 0 IF NO TRANSFER  
ADDRESS IN RECORD
- = 1 IF TRANSFER ADDRESS  
PRESENT

EXPLANATION

RECORD LENGTH = 4 WORDS  
IDENT = 101

8100-10

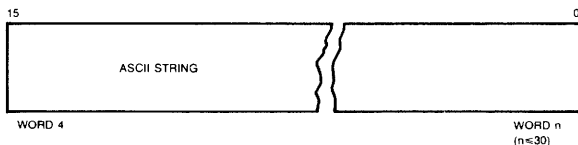
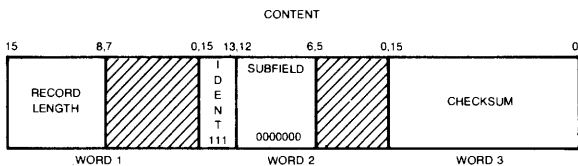
EXTENDED END RECORD (XEND)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	RECORD LENGTH								<ZERO>							
2	IDENT = 7			SUB-IDENT = 5					<ZERO>							
3	CHECKSUM															
4	CHECKSUM OF CHECKSUMS															
5	<ZERO>												TRANS TYPE			
6	0	SYMBOL ID NUMBER										MR				
7	UNRELOCATED VALUE OF OFFSET															
8																

8100-23

# TABLES

## GEN RECORD

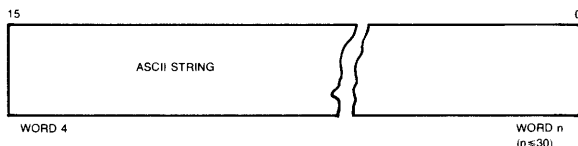
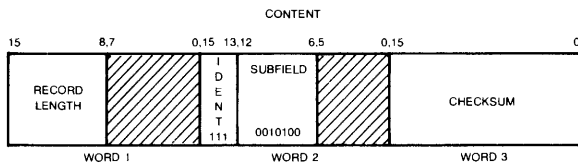


### EXPLANATION

RECORD LENGTH = 3-30 WORDS    ASCII STRING UP TO  
 IDENT = 111                            27 WORDS OR 54 CHARACTERS.  
 SUBFIELD = 0

8100-25

## LOD RECORD



### EXPLANATION

RECORD LENGTH = 3-30 WORDS    ASCII STRING UP TO  
 IDENT = 111                            27 WORDS OR 54  
 SUBFIELD = 24B                            CHARACTERS

8100-26

DATA RECORD

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	RECORD LENGTH								<ZERO>							
2	IDENT 7				SUB-IDENT 7				<ZERO>							
3	CHECKSUM															
4	0	SYMBOL ID NUMBER										MR				
5	UNRELOCATED VALUE OR OFFSET															
6	UNRELOCATED VALUE OR OFFSET															
7	NO. WORDS TO RELOCATE															
8	NO. WORDS TO RELOCATE															
9	FIRST REPETITION COUNT															
10	FIRST REPETITION COUNT															
11	<ZERO>								NO. WORDS IN PATTERN							
	WORDS OF PATTERN (VARIABLE LENGTH)															
	SECOND REPETITION COUNT															
	<ZERO>								NO. WORDS IN PATTERN							
	WORDS OF PATTERN (VARIABLE LENGTH)															

8100-28

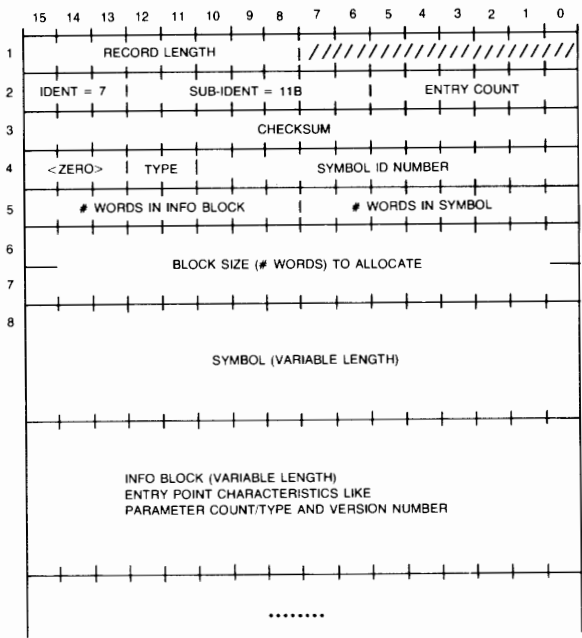
**TABLES**

**RPL RECORD**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	RECORD LENGTH								<ZERO>							
2	IDENT = 7				SUB-IDENT = 6						<ZERO>					
3	CHECKSUM															
4	<ZERO>								NO. WORDS TO REPLACE							
5	WORDS IN INFO FIELD								WORDS IN SYMBOL							
REPLACEMENT VALUE (VARIABLE LENGTH)																
SYMBOL (VARIABLE LENGTH)																
MATCH COND				INFO BLOCK (VARIABLE LENGTH) ENTRY POINT CHARACTERISTICS LIKE PARAMETER COUNT/TYPE AND VERSION NUMBER												

8100-41

## ALLOCATE RECORD



///// MEANS ZERO-FILLED WHEN RECORD IS GENERATED.

- TYPE = 0, IF NAMED COMMON (PROGRAM ALLOCATE)  
 1, IF NAMED SAVE COMMON (SAVE ALLOCATE)  
 2, IF NAMED EMA COMMON (EMA ALLOCATE)

### EXPLANATION

REC LENGTH < 128 WORDS

CHECKSUM: ARITHMETIC  
 TOTAL OF ALL WORDS IN  
 RECORD EXCEPT 1 AND 3

8100-43



**TABLES**

**MSEG RECORD**

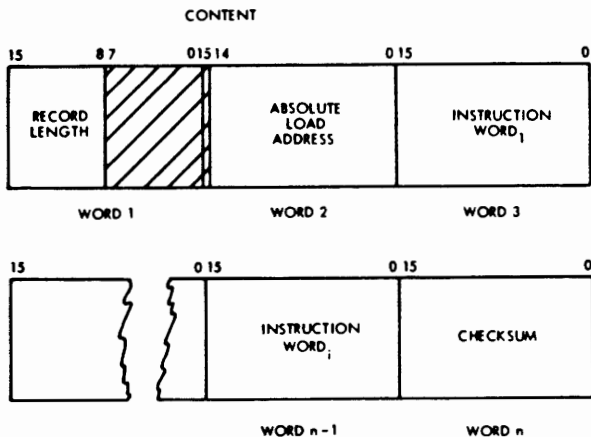
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	RECORD LENGTH								<ZERO>							
2	IDENT = 7			SUB-IDENT = 10B						<ZERO>						
3	CHECKSUM															

8100-42

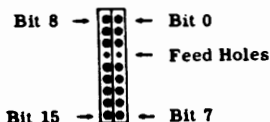
1 ≤ MSEG SIZE ≤ 32

## ABSOLUTE TAPE FORMAT

Absolute binary code is written to paper tape in the following format:



Each word represents two frames arranged as follows:



## EXPLANATION

RECORD LENGTH = NUMBER OF WORDS IN RECORD EXCLUDING WORDS 1 AND 2 AND THE LAST WORD.

ABSOLUTE LOAD ADDRESS: STARTING ADDRESS FOR LOADING THE INSTRUCTIONS WHICH FOLLOW

INSTRUCTION WORDS: ABSOLUTE INSTRUCTIONS OR DATA

CHECKSUM: ARITHMETIC TOTAL OF ALL WORDS EXCEPT FIRST AND LAST

TABLES

FMGR GLOBAL EQUIVALENCE

S	G	P
0	-2	-48 Type
		-47 1
		-46 2
		-45 3
1	-1	-44 Type
		-43 1
		-42 2
		-41 3
2	0	-40 Type
		-39 1
		-38 2
		-37 3
3	1	-36 Type
		-35 1
		-34 2
		-33 3
4	2	-32 Type
		-31 1
		-30 2
		-29 3
5	3	-28 Type
		-27 1
		-26 2
		-25 3
6	4	-24 Type
		-23 1
		-22 2
		-21 3
7	5	-20 Type
		-19 1
		-18 2
		-17 3
8	6	-16 Type
		-15 1
		-14 2
		-13 3
9	7	-12 Type
		-11 1
		-10 2
		- 9 3
10	8	- 8 Type
		- 7 1
		- 6 2
		- 5 3
11	9	- 4 Type
		- 3 1
		- 2 2
		- 1 3
12	10	0 Type
		1 1
		2 2
		3 3
13	11	4 4
		5 5
		6 6
		7 7
		8 8
		9 9

Last FMGR error  
 Severity code  
 Session identifier  
 User's capability level

The standard values are shown within dark lines.  
8100-3

## GENERAL WAIT STATE MESSAGES (State 3)

MESSAGE	REASON FOR WAIT
LULK lu, LKPRG= progx	The listed program attempted to put a lock on logical unit lu. Program progx already has a lock on lu. The listed program will be rescheduled when progx removes its lock.
RN xx, LKPRG= progx	The listed program attempted to set resource number xx. Program progx already has a lock on the resource number. The listed program will be rescheduled when progx removes the lock.
RESOURCE	The listed program attempted to allocate a resource number. The system has no more resource numbers available. The operating system will reschedule the listed program when a resource number is available.
CLASS #	The listed program requested a class number but the system has no more available. The operating system will reschedule the listed program when a class number becomes available.
CL xx	The listed program is waiting on completion of a class GET to class number xx.
progx	The listed program scheduled progx with wait. The listed program will be rescheduled when progx completes.
progx's QUEUE	The listed program scheduled progx on the queue with wait. progx is not dormant so the listed program must wait. The listed program will be rescheduled after the scheduling of progx completes.
BL,EQT xx	Buffer limit exceeded on the controller in EQT entry xx.
EQLK xxx, LKPRG= PRGA	Program suspended for a locked EQT.
EQLK TABLE FULL	Program attempts to lock an EQT and the EQT table is full.



# ERROR CODES

CONTENT	PAGE
ACCOUNT .....	N-2
CLRQ .....	N-4
CMD .....	N-5
COMPL,CLOAD .....	N-6
DISC ALLOCATION .....	N-8
EXEC CALL .....	N-8
FMGR .....	N-9
FMGR UNNUMBERED .....	N-14
GASP .....	N-15
GENIX .....	N-16
INDXR .....	N-17
I/O CALL .....	N-19
LIBRARY .....	N-22
LOADR/MLLDR .....	N-24
LOGON .....	N-29
LU LOCK .....	N-29
MACROASSEMBLER .....	N-30
OUTSPOOL .....	N-40
PARITY ERRORS .....	N-41
READT/WRITT .....	N-42
RECONFIGURATION .....	N-45
RESOURCE NUMBER .....	N-46
SCHEDULE CALL .....	N-46
SCOM .....	N-48
SMP .....	N-49
SYSTEM AND BREAKMODE .....	N-50
SYSTEM BOOT-UP HALTS .....	N-51
TRACK ERROR .....	N-52
VMA/EMA .....	N-53
CI .....	N-54
FMP .....	N-54

## ERROR CODES

### ACCOUNT ERROR CODES

ACCT-225	Session memory can not be returned to system (reboot)	☾
ACCT-223	Illegal shut down parameter	
ACCT-222	Illegal system lu	
ACCT-221	Not an active session	☾
ACCT-220	Corrupt station table spares	
ACCT-219	Not enough room in file for new table	
ACCT-218	Session not shut down	
ACCT-216	Illegal response for primary program.	☾
ACCT-215	List NAMR in transfer stack	
ACCT-213	Invalid memory request	
ACCT-212	Invalid number of SST spares	
ACCT-211	User or group ID not available	
ACCT-210	Conflict in SST definition	
ACCT-209	Invalid SST entry	
ACCT-208	Invalid disc limit	
ACCT-207	Invalid capability	
ACCT-206	Invalid file name	
ACCT-205	Invalid command	☾
ACCT-204	Invalid password	
ACCT-203	Invalid account name	
ACCT-202	Account with this name already exists	☾
ACCT-201	No free accounts	
ACCT-200	Account not found	
ACCT-099	An Exec request made by D.RTR was aborted.	☾
ACCT-046	Attempt to create extent 256. Make file size of main larger.	☾
ACCT-041	No room in SST	

ACCT-040	Lu not found in SST
ACCT-039	Conflict in SST definition
ACCT-035	Already 63 discs mounted to system
ACCT-034	Disc already mounted.
ACCT-033	Not enough room on cartridge
ACCT-032	Cartridge not found
ACCT-030	Value too large for parameter
ACCT-026	Queue full or max pending spools exceeded
ACCT-025	No SPLCON room; the SPLCON is full
ACCT-024	No more batch switches
ACCT-023	No available spool files
ACCT-022	No available spool lu's
ACCT-021	Illegal destination lu
ACCT-020	Illegal access lu
ACCT-019	Illegal access on a system disc
ACCT-018	Illegal lu; lu not assigned to system
ACCT-017	Illegal read/write on Type 0 file
ACCT-016	Illegal Type 0 or file blocks size=0
ACCT-015	Illegal name
ACCT-014	Directory full
ACCT-013	Disc locked
ACCT-012	EOF or SOF error
ACCT-011	DCB not open
ACCT-010	Not enough parameters



## ERROR CODES

ACCT-009	Attempt to use APOSN or force a Type 0 file to Type 1	
ACCT-008	File open or lock rejected	☾
ACCT-007	Illegal security code or illegal write on lu2 or 3	
ACCT-006	File not found	☾
ACCT-005	Record length illegal	
ACCT-004	More than 32767 records in a Type 2 file	
ACCT-003	Backspace illegal	
ACCT-002	Duplicate file name	☾
ACCT-001	Disc error	
ACCT 004	Illegal lu	
ACCT 012	Lu not in session switch table	
ACCT 013	Transfer stack overflow	
ACCT 046	Insufficient capability	

## CLRQ ERROR CODES

CL01	Illegal class number or no class table in system	
CL02	Parameter or calling-sequence error	☾




**CMD ERROR MESSAGES**


☾	ERROR MESSAGE	MEANING
	FMGR ERROR - xxx ON FILE YYYYYY	FMGR error occurred.
	☾	CMD CANNOT FIND YOUR HELP FILE. NOTIFY SYSTEM MANAGER.
	CORRUPT HELP FILE 1) MAKE SURE HELP FILE CREATED BY GENIX 2) MAKE SURE HELP FILE IS TYPE 1 BINARY 3) NOTIFY SYSTEM MANAGER	Input file may not have been indexed by GENIX or may have been corrupted.
	☾	HELP FILE CURRENTLY IN USE BY ANOTHER PROGRAM OR PROGRAMS.
	SECURITY CODE FOR HELP FILE NOT VALID. NOTIFY SYSTEM MANAGER.	Security code viola- tion for access for file has occurred.
	☾	PARAMETER ERROR LIST LU GIVES AN ERROR ON WRITE. GIVE VALID LU FOR SECOND PARAMETER OR DEFAULT TO YOUR TERMINAL
	☾	A bad parameter was entered. This may oc- cur if the list lu is a disc or is non-existent.
	☾	


## ERROR CODES


### COMPL AND CLOAD ERROR CODES


- |        |  |   |
|--------|--|---|
| CL- 01 | The input to the COMPL & CLOAD programs must be a source file.   | ☾ |
| CL- 02 | An FMP error was detected on the open request.   |   |
| CL- 03 | An FMP read error occurred.  | ☾ |
| CL- 04 | An FMP error was detected on the close request.  |   |
| CL- 05 | Control statement not in first 10 lines of the program.  | ☾ |
| CL- 06 | The language requested was rejected by the operating system. The language was purged from the system between the 'RP' and the EXEC request.  |   |
| CL- 07 | The language requested in the control statement was recognized but not found.  |   |
| CL- 08 | The language requested exists on the system and COMPL or CLOAD was in the process of 'RP'ing it. When the file was closed an FMP error occurred.   |   |
| CL- 09 | The language requested exists on the system and COMPL or CLOAD was in the process of 'RP'ing it. However, that 'RP' failed because the checksum calculated when the language was 'SP'ed did not match the system checksum. | ☾ |
| CL- 10 | The language requested exists on the system and COMPL or CLOAD was in the process of 'RP'ing the language. However, during the open request an FMP error occurred.   | ☾ |
|        |  | ☾ |


- 

CL- 11            This session has more than 80 spool files currently residing on the spool disc.
- CL- 12            The compiler was aborted.
- CL- 13            The compilation was not successful. Errors or warnings were found.
- 

CL- 14            This error results when the system is out of ID segments and it is impossible to 'RP' the compiler or LOADR.
- CL- 15            This error means that one of the input parameters was in error.
- 

CL- 30            CLOAD was trying to 'RP' the LOADR but encountered an FMP error on the close of the file that contained the LOADR.
- CL- 31            CLOAD was trying to 'RP' the LOADR and a checksum error resulted.
- CL- 32            CLOAD was trying to 'RP' the LOADR but encountered an FMP error on the FMP open request.
- CL- 33            If the LOADR was not loaded at generation time or an illegal non supported memory or disc modification has been made.
- 



CL- 34            The LOADR was loading your program but was aborted abnormally.
- CL- 35            The load was not successful.
- 

CL- 36            CLOAD was unable to create a copy of the LOADR and even the original LOADR was not available.
- 




CL- 37            The list device for CLOAD must be an lu because both the compiler and the LOADR must talk to the device.

## ERROR CODES

### DISC ALLOCATION ERROR CODES

- DR01 Not enough parameters were specified. 
- DR02 The number of tracks is  $\leq$  zero or an illegal logical unit was specified.
- DR03 An attempt to release a track assigned to another program was made. 

### EXEC CALL ERROR CODES

- DM Mapping error. An attempt was made to read/write outside of the mapped address space. The error message is:  
DM VIOL = wwwwww (DMS violation register)  
DM INST = xxxxxx (offending instruction code)  
ABE pppppp qqqqqq r (A,B, and E registers)  
XYO pppppp qqqqqq r (X,Y, and O registers)  
DM program O 
- MP Memory protect error. The call was not an EXEC, \$LIBR, or \$LIBX call. The following message results:  
MP INST = xxxxxx (offending instruction code)  
ABE pppppp qqqqqq r (A,B, and E registers)   
XYO pppppp qqqqqq r (X,Y, and O registers)  
MP program O 

The following errors have the same format as the DM or MP errors except that register contents are not reported.

- RE            A re-entrant subroutine attempted to call itself.
- RQ            An illegal request code is specified in an EXEC call.
- TI            A batch program exceeds the allowed time.

### **FMGR ERROR CODES**

- FMGR-105    D.RTR directory track buffer too small
- FMGR-104    Requested extent is missing
- FMGR-103    File directory is corrupt
- FMGR-102    Illegal D.RTR call sequence
- FMGR-101    Illegal parameter in D.RTR call
- FMGR-099    Directory manager EXEC request was aborted
- FMGR-052    Spool shut down. Spool file setup failed
- FMGR-049    Copy verify failed
- FMGR-048    Spool not initialized or SMP cannot be scheduled
- FMGR-047    No session lu available for spool file
- FMGR-046    Greater than 255 extents
- FMGR-041    No room in SST
- FMGR-040    Lu not found in SST
- FMGR-039    Spool lu not mapped to the spool driver
- FMGR-038    Illegal scratch file number
- FMGR-037    Attempt to purge an active type 6 file
- FMGR-036    Lock error on device

## ERROR CODES

FMGR-035	Already 63 discs mounted to system	
FMGR-034	Disc already mounted.	☾
FMGR-033	Not enough room on cartridge	
FMGR-032	Cartridge not found	
FMGR-030	Value too large for parameter	
FMGR-026	Queue full or max pending spools exceeded	☾
FMGR-025	No SPLCON room	
FMGR-024	No more batch switches	
FMGR-023	No available spool files	☾
FMGR-022	No available spool lu's	
FMGR-021	Illegal destination lu	
FMGR-020	Illegal access lu	
FMGR-019	Illegal access on a system disc	
FMGR-018	Illegal lu	
FMGR-017	Illegal read/write on Type 0 file	
FMGR-016	Illegal Type 0 or size=0	
FMGR-015	Illegal name	
FMGR-014	Directory full	
FMGR-013	Disc locked	☾
FMGR-012	EOF or SOF error	
FMGR-011	DCB not open	
FMGR-010	Not enough parameters	☾
FMGR-009	Attempt to use APOSN or force to 1 a Type 0 file	
FMGR-008	File open or lock rejected	
FMGR-007	Illegal security code or illegal write on lu2 or 3	☾
FMGR-006	File not found	

- FMGR-005 Record length illegal
- FMGR-004 Record size of Type 2 file is 0 or undefined
- FMGR-003 Backspace illegal
- FMGR-002 Duplicate file name
- FMGR-001 Disc error, the disc is down.
- FMGR 000 Break, informative message only no error has occurred.
- FMGR 001 Disc error — lu reported, disc associated with the lu is down.
- FMGR 002 Initialize lu 2!
- FMGR 003 Initialize lu 3!
- FMGR 004 Illegal response to FMGR 002 or FMGR 003
- FMGR 005 Required track not available — relative TAT position reported
- FMGR 006 FMGR suspended
- FMGR 007 Checksum error
- FMGR 008 D.RTR not loaded
- FMGR 009 ID segment not found
- FMGR 010 Input error
- FMGR 011 Do 'OF,XXXXX,8' on named programs
- FMGR 012 Duplicate disc label or lu
- FMGR 013 TR stack overflow
- FMGR 014 Required ID segment not found
- FMGR 015 LS track report
- FMGR 016 Insufficient system tracks for RP
- FMGR 017 ID segment not set up by RP
- FMGR 018 Program not dormant
- FMGR 019 File not set up by SP on current system



## ERROR CODES

FMGR 020	Illegal Type 0 file	
FMGR 021	Illegal disc specified	☾
FMGR 022	Copy terminated	
FMGR 023	Duplicate program name	
FMGR 038	Attempt to purge active file	
FMGR 041	Program cannot be a segment	☾
FMGR 042	Lu cannot be switched	
FMGR 043	Lu not found in SST	
FMGR 044	No messages waiting	
FMGR 045	Session command only	☾
FMGR 046	Insufficient capability	
FMGR 047	Spool set up failed	
FMGR 048	Global set out of range	
FMGR 049	Can't run RP'ed program	
FMGR 050	Not enough parameters	
FMGR 051	Illegal master security code	
FMGR 052	Illegal lu	
FMGR 053	Illegal label or ilabel	
FMGR 054	Disc not mounted	☾
FMGR 055	Missing parameter	
FMGR 056	Bad parameter	
FMGR 057	Bad track not in file area	
FMGR 058	LG area empty	☾
FMGR 059	Reported track unavailable	
FMGR 060	Do you really want to purge this disc?	
FMGR 061	Do a "DC" and a "MC" on this CR	
FMGR 062	More than 63 discs	☾

- FMGR 063 Exceeding session disc limit
- FMGR 064 No disc available from disc pool.
- FMGR 065 Conflict in SST definition
- FMGR 066 No room in SST
- FMGR 067 Program not found
- FMGR 068 Lu not in variable part of SST
- FMGR 069 Job LOGON failed
- FMGR 070 Sectors/track value too large
- FMGR 071 Do "EX,SP" to save or "EX,RP" to release private cartridges
- FMGR 072 Lu not interactive
- FMGR 073 Account not found
- FMGR 074 JO command expected
- FMGR 075 Can't restore Type 6 PGM file (user protected)
- FMGR 076 Can't restore Type 6 PGM file (group protected)
- FMGR 077 Can't restore Type 6 PGM file (insufficient capability)
- FMGR 078 Cannot restore Type 6 program file (internal error)
- FMGR 079 Warning — records truncated to 128 words
- FMGR 080 Cannot find EMA in system

## ERROR CODES

### FMGR UNNUMBERED

ERROR

MESSAGE    MEANING

**ABEND OPERATOR**    The job has been aborted by operator request, or has been aborted because of spool I/O error.

**JOB xxxxx ABORTED**    Error encountered during job execution.

**ABEND EOJ IN ssssss**    An :EO or :JO command was encountered, but in a different level from the original :JO command. For example, control has transferred from PROG1 to PROG2. PROG2 contains :EO or :JO command. ssssss is the file name or logical unit number where :EO or :JO occurred.

**ABEND JOB LIMIT**    The job time limit (set via the :JO command) has been exceeded.

**ABEND RUN LIMIT**    The run time limit (set via the :TL command) has been exceeded.

**FMGR WAITING ON LU xx**    LU xx is down or locked to another program.

**GASP ERROR CODES**

- ☾ GASP -48 Spooling not initialized or SMP cannot be scheduled
- GASP -33 Not enough room on cartridge
- GASP -32 Cartridge not found
- ☾ CASP -14 Directory full
- GASP -13 Disc locked
- GASP -12 EOF or SOF error
- GASP -8 File open or lock rejected
- ☾ GASP -7 Illegal security code or illegal write on lu2 or 3
- GASP -6 File not found or no room to create spool files
- GASP -4 More than 32767 records in a Type 2 file
- GASP -2 Duplicate file name
- GASP -1 Disc error, disc is down
- GASP 1 Disc associated with lu NN is down
- GASP 2 Number out of range
- GASP 3 Bad job number!
- ☾ GASP 4 Illegal status
- GASP 5 Illegal command
- GASP 6 Not found. Specified job or spool not currently assigned
- ☾ GASP 7 GASP segment not found
- GASP 43 Lu not found in SST
- GASP 46 Insufficient capability
- ☾ GASP 54 Spool cartridge not mounted
- GASP 55 Missing parameter
- GASP 56 Bad parameter

## ERROR CODES

### GENIX ERROR MESSAGES

ERROR MESSAGE	MEANING	
HEAP/STACK COLLISION IN LINE xxxx	Too many keys to save in the Pascal heap area.	☾
NO KEYWORDS FOUND	The input file contains no key words.	☾
WARNING: BLANK KEYWORD FOUND	A block of text, pre- ceded with a blank key word, is found.	☾
DUPLICATE KEYWORD 'xxx...x'	A duplicate key word is found.	☾
REGULAR PASCAL I/O ERRORS	FMP and I/O errors are handled using the stand- ard Pascal methods.	☾
		☾
		☾
		☾
		☾

**INDXR ERROR MESSAGES**

**ERROR  
MESSAGE**

**MEANING**

??

An invalid command was entered, try again.

**\*\*\* CAUTION ! INDEXED  
FILE ALREADY EXISTS,  
OVERLAY(Y OR N)? \*\*\***

INDXR has found in response to a "CR" command that the indexed file specified already exists.

**\*\*\* INPUT ERROR, INDXR  
ABORTED \*\*\***

An invalid transfer file was specified in the run string.

**\*\*\* ONLY ONE LEVEL  
OF TRANSFER  
ALLOWED \*\*\***

Only one transfer file may be opened for command input to this utility.

**\*\*\* INDXED FILE ALREADY  
EXISTS, REQUEST  
IGNORED \*\*\***

An attempt was made to use the CReate command more than once.





**\*\*\* FILE MUST BE TYPE 5,  
REQUEST IGNORED \*\*\***

The file specified in the INdex command was not a type five (relocatable) file.

**\*\*\* CHECKSUM ERROR,  
INDXR ABORTED \*\*\***

While reading records from the file specified by the last INdex command, a checksum error was detected in one of the records.

## ERROR CODES

- |  |   |   |
|--|---|---|
| *** DIRECTORY TOO<br>LARGE, INDXR<br>ABORTED ***         | The INDXR creates a<br>scratch file to build the<br>directory index in. This<br>file is created on the<br>first cartridge in the<br>cartridge list. If this file<br>creates more than 255<br>extents or fills up the re-<br>mainder of the cartridge,<br>then this error is returned. |    |
| *** SCRATCH FILE<br>CREATE<br>OVERFLOW ***               | The INDXR has tried to<br>create a scratch file with<br>a name in the range from<br>@DIR@A to @DIR@Z,<br>but has found that all<br>these files already exist.   |    |
| *** LIST FILE MUST BE<br>TYPE 3 OR 4 ***                 | An already existing FMP<br>file used as a list file<br>must be a type 3 or 4.   |   |
| *** LIST FILE/LU ALREADY<br>OPEN, REQUEST<br>IGNORED *** | The list file may only be<br>opened/created once.<br>If the LI[ST] command<br>is invoked more than<br>once then this error is<br>issued.  |  |

**I/O CALL ERROR CODES**

IO00 An illegal class number was specified. Outside table, not allocated, or bad security code.

IO01 Not enough parameters were specified.

IO02 An illegal logical unit number was specified.

IO03 Illegal EQT referenced by lu in I/O call (select code=0).

IO04 An illegal user buffer was specified. Extends beyond RT/BG area or not enough system available memory to buffer the request.

IO05 An illegal disc track or sector was specified.

IO06 A reference was made to a protected track or to unassigned LG tracks.

IO07 The driver has rejected the call.

IO09 The LG tracks overflowed.

IO10 Class get call issued while one call already outstanding.

IO11 A Type 4 program made an unbuffered I/O request to a driver that did not do its own mapping.

IO12 An I/O request specified a logical unit not defined for use by this session. The format for IO12 error is:

SES LU = XX







IO12 PROG ADDRESS

Where:

XX = session lu not in SST



## ERROR CODES

IO13	An I/O request specified an lu which was either locked to another program, or pointed to an EQT which was locked to another program.	
IO14	An I/O request was issued with the no-suspend option.	
IO15	Buffer size of a type 6 program is greater than what will fit in the user map.	
IO16	CPU backplane failure or I/O extender timing failure.	
IO20	Read attempted on write only spool file.	
IO21	Read attempted past end-of-file.	
IO22	Second attempt to read JCL card from batch input file by other than FMGR. Revise program and re-run.	
IO23	Write attempted on read only spool file.	
IO24	Write attempted beyond end-of-file; usually, spool file overflow.	
IO25	Attempt to access spool lu that is not currently set up.	
IO26	I/O request made to a spool that has been terminated by the GASP KS command.	
		
		

ILL INT

An illegal interrupt occurred on the specified channel.

The following error message format is used to report I/O errors:

```
NR
IOET L xxx E yyy S zz qqq
TO
PE
```

Where:

xxx = device's lu

yyy = device's EQT

zz device's subchannel

qqq= device status returned by driver  
(if the driver is down at I/O request the status=\*\*\*).

IOET

An end-of-tape condition occurred on the specified lu.

IONR

The specified lu is not ready. Make the device ready and set the EQT up.

IOTO

The specified lu has timed out.

IOPE

A parity error occurred in the data transmission from the specified lu.

## ERROR CODES

### LIBRARY ERRORS

#### Mathematical Subroutines

OF = Integer or Floating Point Overflow

OR = Out of Range

UN = Floating Point Undefined

Error Message	Issuing Subroutine	Where Used	Error Condition
02-UN	ALOG	ALOG ALOGT CLOG	$X \leq 0$ $X \leq 0$ $x = 0$
03-UN	SQRT	SQRT DSQRT	$X < 0$
04-UN	.RTOR	.RTOR	$X = 0, Y \leq 0$ $X < 0, Y \neq 0$
05-OR	SIN	SIN CSNCS CEXP COS	$\frac{1}{2} \left  \frac{X}{\pi} + \frac{1}{2} \right  > 2^{14}$
06-UN	.RTOI	.RTOI	$X = 0, Y \leq 0$
07-OF	EXP	EXP CEXP .RTOR CSNCS	$X * \log_2 e \geq 124$ $X_1 * \log_2 e \geq 124$ $ X * \text{ALOG}(X)  \geq 124$ $X_2 * \log_2 e \geq 124$
08-UN	.ITOI	.ITOI	$I = 0, J \leq 0$
08-OF	.ITOI	.ITOI	$I^J \geq 2^{15}$ or $I^J < -2^{15}$
09-OR	TAN	TAN	$X > 2^{14}$
10-OF	DEXP	DEXP .DTOD .DTOR .RTOD	$e^X > (1-2^{-39}) 2^{127}$ $X > (1-2^{-39}) 2^{127}$

11-UN	DLOG	DLOG	$X \leq 0$
		DLOGT	$X < 0$
12-UN	.DIOI	.DIOI	$X = 0, I \leq 0$
13-UN	.DTOD	.DTOD	$X = 0, Y \leq 0$
		.DTOR	$X < 0, Y = 0$
		.RTOD	
14-UN	.CTOI	.CTOI	$X = 0, I \leq 0$
15-UN	DATN2	DATN2	$X = Y = 0$

Utility Subroutines

Subroutine

Error

MAGTP







Returns on an illegal call.


.SWCH


Returns if element is out of range.


## ERROR CODES


### LOADR/MLLDR ERROR CODES


- |          |  |   |
|----------|--|---|
| L-CK SUM | This is a checksum error. A file specified to the loader that did not contain relocatable format code.   |    |
| L-CM BLK | This is a common block error.  |   |
| L-CO RES | Attempt to replace or purge a memory-resident program. This is illegal.  |    |
| L-DU ENT | Duplicate entry point.   |   |
| L-DU PGM | Attempt to load the same program several times without getting rid of the earlier loads.   |    |
| L-EX CPY | Attempt to replace or purge a program where copies of that program existed.  |   |
| L-ID EXT | No ID extensions available for the EMA program.  |   |
| L-IL ALC | External references to named COMMON which appear in an allocate record are not allowed before the allocate record occurs.  |   |
| L-IL CMD | Attempt to purge a program under batch or attempt to use the PU command within a loader command file.  |  |
| L-IL DRN | Illegal disc-resident node specification.  |   |
| L-IL EMA | Tried to use shareable EMA with the old EMA relocatable format.  |   |
| L-IL MLS | This error occurs during preliminary checking of the command file, or when an instruction other than a JSB is used to access a symbol in a son node during load. |  |
| L-IL PRM | The runstring or a command in a command file contained an error.   |  |


- 


**L-IL PTN**      A partition specified in the load of the program, does not exist or has been downed due to a parity error.
- 


**L-IL REC**      The loader found a record that was not a NAM, ENT, EXT, DBL, EMA, GEN, LOD, END record, or extended record. The checksum was OK but the record was not identified.
- 


**L-IL REL**      The compiler produced an illegal record.
- 

**L-IL RPL**      Tried to do a JSB to a user-specified RPL in a son node.
- 

**L-IL SCB**      Illegal session control block value (negative capability level).
- 

**L-IL SEG**      Illegal segment specification.
- 

**L-IN CAP**      Attempt to load, purge, or replace a permanently loaded program without having a session capability level high enough to perform this function.
- 

**L-LM LIB**      The limit on the number of libraries specified by the 'LI' command has been exceeded. You may specify 10 libraries.
- 

**L-ML BDT**      Multiple block data subprogram. Attempted to initialize the same area more than once.
- 

**L-ML EMA**      Illegal EMA declaration.
- 







**L-NO IDS**      Not enough ID segments to finish the load.
- 


**L-NO RSG**      No root segment specified.
- 


**L-NO SNP**      Notify your system manager. The loader could not find the file \$\$SYENT.
- 


**L-OV BSE**      Base page overflow.


## ERROR CODES


- L-OV DSK Program exceeds the maximum disc space allowed a program. 
- L-OV FIX This is a fixup table overflow.
- L-OV MEM The relocation address has exceeded 77777B, 77777B — MSEG, or the size specified using "SZ,N". 
- L-OV PTN This size specified using the SZ command was too large for the program's type.
- L-OV RBP Overflow of rotating base page (MLLDR only). 
- L-OV SAV Overflowed SAVE area.
- L-OV SNP Overflowed snap file \$SYENT. Try running the loader using RU, LOADR, -1, -1 to create the \$SYENT file.
- L-OV SYM This is a symbol table overflow.
- L-PE LDR Tried to do a purge, replace, or permanent load with a copy of the loader.
- L-RE SEQ Record out of sequence.
- L-RF EMA Attempt to access an EMA external with offset or indirect.
- L-RP CPY Attempt to replace a copied program. 
- L-RP MLS Tried to use MLLDR to replace a program that was loaded by LOADR or vice-versa. Another cause is trying to replace or purge a memory-resident program using MLLDR. 
- L-RP PGM Tried to replace or purge a permanent program that has terminated serially reusable, saving resources, or was operator suspended. 
- L-RQ PGS Both a SZ and AS were used and the size is larger than the partition.


- 

**L-SH EMA**    A shareable EMA label file did not have the control command \$SHEMA starting in the first column in the first line, or there was another error in the file.
- 

**L-SH PTN**    A program cannot be assigned to a shareable EMA partition, a subpartition of a shareable EMA partition, or a mother partition which has any shareable EMA subpartitions.
- 

**L-SS ENT**    Attempt to access an SSGA entry point without asking for SSGA at the beginning of the load. Reload the program but this time do an 'OP,SS' at the beginning of the load.
- L-SZ ALC**    Allocate size error.
- L-SZ EMA**    EMA size is greater than 1K pages and VMA is not being used.
- L-TR ADD**    No transfer address. Only subroutines were loaded.
- L-UN EXT**    Undefined externals exist which prohibits the load from completing.
- 

**L-VM EMA**    Tried to use shareable VMA (not supported).
- L-VS EMA**    The specified VMA size is illegal.
- W-DU PGM**    Duplicate program name.
- 

**W-IL CMD**    Attempted to relocate a module or transfer to a command file while doing special processing when undefined externals exist.
- 



## ERROR CODES

- W-IN CAP** Due to insufficient user capability, the program priority was changed to 99. The message is:  
/MLLDR: XXXXX SET TO PRIORITY 99  
/MLLDR: W-IN CAP
- W-RQ PGS** The required number of pages is too large.
- W-SV MIX** Mixing SAVE named COMMON with named COMMON.
- W-UN EXT** Undefined externals exist and the loader was initiated from an interactive device.
- W-WS EMA** The specified working set size is too large.



**LOGON ERROR CODES**

- LGON 00 Session environment not initialized
- LGON 01 FMP error on account file access
- LGON 03 Session limit exceeded
- LGON 04 No such user
- LGON 05 Illegal access
- LGON 06 Conflict in definition of session lu
- LGON 07 No room for session control block
- LGON 08 Duplicate session identifier
- LGON 09 SST overflow
- LGON 10 No free ID segments or FMGR not found
- LGON 11 FMP error on disc mount attempt
- LGON 12 Account file corrupt
- LGON 13 Conflict with system disc lu
- LGON 14 Bad job Log-on request
- LGON 15 Session Primary Program not found

**LU LOCK ERROR CODES**

- LU01 A program has one or more logical units locked and is trying to lock another with wait.
- LU02 Illegal logical unit reference.
- LU03 Not enough parameters in the call; lu reference is less than one; or lu not locked to caller.
- LU04 Trying to lock a logical unit not defined in caller's SST.

## ERROR CODES

### MACROASSEMBLER ERROR CODES

ERROR CODE	EXPLANATION	
1	Illegal file namr	☾
2	Include files may not be nested past 5 deep.	
4	Opcode illegal in absolute assembly.	☾
5	Greater than 1/4 million symbols used. Can't give symbol table dump.	
51	Expression in AIF, or AELSEIF statement does not result in a 0 or 1.	☾
52	End of file found before AENDIF in AIF statement.	
53	AELSE found before AIF. This line gets ignored.	
54	AENDIF found outside of AIF statement. This line gets ignored.	
55	AELSEIF found after AELSE. This line gets ignored.	
56	Only one AELSE allowed per AIF statement. This line gets ignored.	☾
57	Illegal use of AELSEIF. This line gets ignored.	☾
58	AIFs nested past 16 deep. This line gets ignored.	
59	IFNs or IFZs may not be nested. This line gets ignored.	☾
61	XIF found outside of IFN/IFZ statement. Line ignored.	
62	No corresponding MACRO, REPEAT or AWHILE.	☾

ERROR CODE	EXPLANATION
63	Illegal to use ENT and RPL to 2 word RPL values.
64	End of file found before AENDWHILE or ENDREP.
101	Assembly time variable or macro parameter has more than 16 characters.
102	Illegal assembly time variable name.
103	Syntax error in assembly time array : &name[dimension,size].
104	ATV array subscript must be integer > 0.
105	Length of string > size specified in ATV array. Truncated.
106	The "count" field in assembly time array must be integer >0.
107	Missing ']' in operand field of assembly time array.
108	Syntax error in operand field of assembly time array declaration.
109	Not enough initial values for assembly time array.
110	Doubly declared assembly time variable name.
111	Label in ISET, IGLOBAL or ILOCAL statement does not start with '&'.
112	Unrecognized '&' variable.
113	ATV used in a ISET or CSET statement has not been defined.
114	ATV is defined as an array but not used as an array.

## ERROR CODES

ERROR CODE	EXPLANATION	
115	Referencing an element outside the dimension defined by ATV array.	☾
116	String longer than maximum specified in declaration. Truncated.	
117	Result of ILOCAL, or IGLOBAL is not an integer, default to 0.	☾
118	ATV array size must be $\leq 80$ and $> 0$ .	
119	Array subscript must be surrounded by square brackets.	☾
120	Array subscript may not itself be an array.	
121	Comparison is not allowed in ATV manipulation.	
122	Type conflict in ISET or CSET statement, value of ATV is unchanged.	
123	Dimension or size of element in ATV array cannot be $\leq 0$ .	
124	ILOCAL or CLOCAL must be declared inside a macro call.	
125	Array subscript must be single integer or integer variable.	☾
126	Size specified in IGLOBAL and ILOCAL is ignored, default to 1 word.	
127	Too many elements in ATV array declaration, rest are ignored.	☾
128	No operand in CGLOBAL/CLOCAL, default to null string.	
151	Illegal column indicator on MACRO statement.	☾
152	Macro name missing from macro definition.	

ERROR CODE	EXPLANATION
153	Macro name may only contain A-Z, a-z, 0-9, or '.'
154	Macro by this name already defined.
155	'ENDMAC' statement missing.
156	String must be $\leq$ 80 character, truncated.
157	Illegal formal macro parameter.
158	Default value too long for listing.
159	Formal parameter must start with '&'
160	Illegal actual macro parameter.
161	Too many parameters for this macro call.
162	Repeats may not be nested more than 5 deep.
163	Expression on REPEAT or REP must have positive integer result.
164	Illegal expression on AWHILE statement.
165	Expression on AWHILE must have less than 80 characters.
166	More than 100 EXTRACT/DELETE macros for this file.
167	May not use both EXTRACT and DELETE following this INCLUDE or MACLIB.
168	Only 5 macro libraries allowed per program.
201	Mnemonic field missing.
202	Line too long after string substitution.
203	Column indicators should be 3 integers separated by commas.
204	Mnemonic field longer than 16 characters.

## ERROR CODES

ERROR CODE	EXPLANATION	
205	END statement missing.	☾
206	Mnemonic column must start past column 1.	
207	Column indicators must leave room for next field.	☾
208	Comment field must start before column 128.	
209	Label longer than 16 characters.	☾
210	Illegal character in label.	
211	Illegal character in opcode field.	
212	Opcode illegal in this type of assembly.	
213	Operand field missing.	
214	Opcode not recognized.	
215	Undefined symbol.	
216	Too many nested parentheses. Limit is 10.	
217	Incomplete expression in operand field.	
218	String encountered in an integer expression, default to 0.	☾
219	RPL label cannot be used in operand field.	
220	'(' or integer must be preceded by an operator.	
221	Syntax error in expression.	☾
222	Integer divide results in overflow.	
223	& variable must follow :L,:S: or :T: operators.	
224	Illegal use of :T: operator.	☾

ERROR CODE	EXPLANATION
225	:NOT: must be followed by a type integer variable.
226	Syntax error in substring :S:[var,var]string.
227	Number in substring must be $\geq 1$ .
228	Length of substring exceeds current length of string.
229	)' encountered without corresponding '('.
230	)' must be preceded by an integer result.
231	Integer exceeds range $-32768$ to $32767$ .
232	Substring construct may not be nested.
233	Substring starting character exceeds string length.
234	Result of expression must be within $0$ to $32767$ .
235	ASCII string in GEN and LOD record must be $\leq 125$ words.
236	Legal string compare operators are = and <>.
237	Line continuation may not start before the operand field.
238	Duplicate label definition.
239	Illegal operator in expression.
240	Operand must be integer or absolute expression.
241	Undefined entry point.
242	Only one operand may be relocatable.
243	Illegal character in expression.



## ERROR CODES

ERROR CODE	EXPLANATION	
244	Result of an EQU expression cannot be indirect.	☾
245	Illegal floating point number construct.	
251	Illegal column indicator in COL statement.	☾
252	Keyword must be ON, OFF, SHORT, MEDIUM, or LONG.	
253	Octal integers may not contain an 8 or 9.	
254	Literals not legal on this opcode.	☾
255	Keyword must be PROGRAM, COMMON, SAVE, CODE, or BASE.	
256	ORR must appear before this ORB, ORG, or RELOC.	
257	ORR found before corresponding ORG or ORB.	
258	Operand must be absolute or relocatable expression.	
259	Variable not found.	
260	Legal literals are =D, =B, =F, =A, =L, =R, and =S.	☾
261	Integer expected.	
262	ASCII string expected.	
263	Label missing.	
264	Doubly defined entry point name.	☾
265	Illegal value for entry point.	
266	Result of expression must be absolute integer value.	
267	Expression contains 2 different externals.	☾

ERROR CODE	EXPLANATION
268	2 consecutive REP statements encountered.
269	End of file encountered following REP statement.
270	Comment field must be separated from operand field by blank or ';'.
271	Expression cannot exist in more than one relocatable space.
272	Label ignored.
273	Syntax error in MIC statement.
274	Duplicate name for MICro code instruction.
275	Duplicate NAM statement.
276	Keyword must be EMA or SAVE.
277	MSEG size must be $\geq 2$ and $\leq 31$ .
278	Syntax error in ALLOC.
279	EMA and ALLOC EMA or MSEG cannot be used in the same program.
280	Duplicate EMA statement.
281	Label longer than 5 character in EMA statement.
282	Number of pages specified or MSEG size out of range in EMA statement.
283	Syntax error in EMA statement.
284	Result in operand field cannot be type RPL, or EMA.
285	Local EMA label may only be used in a DDEF statement.
286	DBL/DBR cannot be indirect.

## ERROR CODES

ERROR CODE	EXPLANATION	
287	Illegal opcode combination.	☾
288	Illegal data in OCT, DEC, DEX, or DEY.	
289	Byte value overflow, must be within -377B to 377B.	☾
290	Not enough parameters in microcode call.	
291	Literals are not allowed in microcode call.	
292	Expression in RAM pseudo op must be between 0 to 377B	☾
293	Result of expression in DDEF cannot be RPL or indirect.	
300	EXT/ENT statement error.	
301	Illegal symbol in EXT/ENT.	
302	Doubly defined entry point.	
303	Illegal character in Alias field.	
304	Illegal character in Info. field.	
305	EXT & ENT may not reference the same symbol.	
306	Info or alias field on ref. to existing symbol.	☾
307	Number of externals exceeds 2047.	
308	Too many parameter types in info field.	
309	I/O select code must be absolute, >0, <64.	
310	COM operand field error.	☾
311	COM allocation must be absolute and greater than zero.	
312	COM statement contains illegal symbol.	
313	COM statement legal only in program relocation space.	☾

ERROR CODE	EXPLANATION
314	RPL names limited to 5 characters until loader enhancements complete.
315	EMA value not allowed here.
316	Operand must be positive, absolute, and less than or equal to 16
317	Subhead parameter must be less than 81 characters.
318	Name used both for label and for external replacement opcode.
319	Illegal program name.
320	Only the '=F' literal is legal on this opcode.
321	Only the =S, =D, =B, =A, =R, and =L literals are legal on this opcode.
322	Comment field on NAM statement may not exceed 73 characters in length.
323	EQUs may not be negative when 'ASMB' is the control statement.
324	BSS, COM, ORG, RELOC, ORB, or machine insts. may not appear before NAM.
325	NAM statement missing.
326	Values on =L literal must be previously defined.

## ERROR CODES

### OUTSPOOL ERROR MESSAGES

MESSAGE	CAUSE	
JOB WAIT ON PT	End-of-Tape occurred between :JO and :EO commands.	☾
JOB WAIT ON SPOOL RESOURCE	Required spool file or logical device cannot be obtained at this time.	☾
JOB WAIT ON EXTENT	Spool file overflows available disc space.	☾
END JOB ABNORM	JOBFIL could not be opened; or other uncorrectable error occurred; or JOB was run before spool initialization.	☾
BAD EOF	Message appears after last line of file. ASCII file outspooling overflowed; or was otherwise incomplete.	☾
		☾
		☾
		☾

**PARITY ERROR MESSAGES**

 HARD PARITY ERROR MESSAGE (reproducible):

PE PG# nnnnn BAD (physical page # of parity error)

 ABE aaaaaa bbbbbb e (A,B, and E registers)

XYO xxxxxx yyyyyy o (X,Y, and O registers)

PE ppppp mmmmmm (program name and logical  
ppppp ABORTED memory address of parity error)

 SOFT PARITY ERROR MESSAGE (not reproducible):

SOFT PE PG# nnnnn (physical page# of parity error)

ABE aaaaaa bbbbbb e (A,B, and E registers)

XYO xxxxxx yyyyyy o (X,Y, and O registers)

PE ppppp mmmmmm (program name and logical  
memory address of parity error)



## ERROR CODES

### READT/WRIIT ERROR CODES

- |          |  |   |
|----------|--|---|
| READ 001 | The requested mag tape unit is down.   | ☾ |
| READ 002 | The mag tape READT is trying to restore contains information in a format not restorable by READT.                                    |   |
| READ 003 | The mag tape unit you wish to use is locked to some process.   | ☾ |
| READ 004 | The parameter describing the desired mag tape unit does not satisfy READT's requirements for a legal mag tape lu.                    |   |
| READ 005 | The desired mag tape unit is off-line.   | ☾ |
| READ 006 | READT rejected the use of the specified disc lu.   |   |
| READ 007 | The driver detected a parity error when reading from the mag tape.   |   |
| READ 008 | The end of tape was reached.   |   |
| READ 009 | The desired cartridge has a file open or the cartridge is locked to another program.   |   |
| READ 010 | You are operating in a nonsession environment. An lu must be specified (negative lu) since there isn't a free disc pool.             | ☾ |
| READ 011 | READT rejected the size (number of tracks) you specified.  |   |
| READ 012 | The routine READT uses to mount a cartridge detected an error.   | ☾ |
| READ 013 | The desired disc lu or the available free lus in the disc pool are not large enough to restore the cartridge that's on the mag tape. |   |
| READ 014 | The FMP tracks on lu 2 or lu 3 (if 3 exists) are not restorable with READT.  | ☾ |

- READ 015 Bad transmission — memory to disc trk xxx sec yyy READT tried to transfer data from memory to a disc lu. During this process a check of the transmission log showed an unexpected value. Run READT again, if it happens once more call your system manager.
- READ 016 Bad transmission — mag tape to memory rec xxx READT detected an error in transmission of data from the mag tape unit into memory. Try reading the tape again. If it happens once more call your system manager.
- READ 017 Internal buffer too small — the tape records are longer than the READT internal buffer. The buffer size must be increased and READT must be reloaded.
- READ 018 READ aborted by user this message is produced when you respond NO to any prompt, or when READT is halted using the BReak command.
- READ 019 Disc error on lu xx, track xxxx — READT encountered an error when reading the listed track of the listed LU.
- READ 020 Verify error on track xxxx — a compare error was encountered when verifying the listed track.
- READ 021 Invalid parameter — an invalid parameter was specified in the READT command runstring. Check the runstring and re-enter the parameter.
- WRIT 001 The device can be enabled.
- WRIT 002 Only the system manager can save system discs.



## ERROR CODES

- WRIT 003 The mag tape you wish to use is locked to some process.
- WRIT 004 The parameter describing the desired mag tape unit does not satisfy READT's requirements for a legal mag tape unit.
- WRIT 005 The desired mag tape unit is off-line.
- WRIT 006 A write ring is required to write information on a mag tape.
- WRIT 007 The driver detected a parity error when reading from the mag tape.
- WRIT 008 The end of tape was reached.
- WRIT 009 The desired cartridge has a file open or the cartridge is locked to another program.
- WRIT 010 The desired cartridge or disc lu could not be found.
- WRIT 011 WRITT rejected the use of the specified disc lu.
- WRIT 012 You cannot save FMP tracks off lu 2 or lu 3 with WRITT.
- WRIT 013 WRITT tried to read data from a disc lu into memory and found the transmission irregular. Run WRITT again, if the situation occurs once more there may be a bad track on that disc lu. Save as much data as you can and notify your system manager.
- WRIT 014 The transmission of data from memory to mag tape may be faulty. Run WRITT again, if it happens once more call your system manager.

**WRIT 016** An error was detected in transmission of data from the magnetic tape to memory. If this error recurs, the tape may be faulty; see the System Manager.

**WRIT 020** A compare error was encountered when verifying the listed track.

**RECONFIGURATION ERROR CODES**

CONFIG  
ERR

MEANING

- 1 Invalid LU number or a bit bucket LU.
- 2 Illegal select code number.
- 3 New select code entered is identical to new select code assigned to disc system console or list device, or else the current select code entered is identical to the old select code for disc, system console or list device (i.e., do not reconfigure that which was already done via the SWTCH register).
- 10 Specified total number of pages outside the range.
- 11 Invalid bad page number.
- 12 Specified SAM extension entry beyond physical memory size due to bad pages.
- 13 Current running total exceeds available pages in block of good memory or exceeds size of mother partition.
- 14 Second parameter of partition definition entry other than RT, BG or S, or else S was entered when a subpartition definition was not expected.

## ERROR CODES

- |    |   |   |
|----|---|---|
| 15 | Third parameter of partition definition entry other than R.   |   |
| 16 | No such program, or the name of a segment was entered or invalid type was entered for partition assignment. | ☾ |
| 17 | Invalid partition number.   |   |
| 18 | Program does not fit in the assigned partition.   | ☾ |
| 19 | Invalid number of pages was entered for program size.   |   |
| 20 | Number of defined partitions already equal to allowed maximum number and more undefined pages remain.       | ☾ |
| 21 | Page requirements of an EMA program cannot be modified.   |   |
| 22 | Number of pages in SAM extension requires division into more than five blocks.                              |   |

## RESOURCE NUMBER ERRORS

- |      |  |   |
|------|--|---|
| RN00 | There are no option bits set in the call.                          |   |
| RN01 | No resource numbers in system.                                     | ☾ |
| RN02 | The specified resource number is not defined.                      |   |
| RN03 | An unauthorized attempt was made to clear a local resource number. | ☾ |

## SCHEDULE CALL ERROR CODES

- |      |   |   |
|------|---|---|
| SC00 | A batch program attempted to suspend (EXEC(7)). |   |
| SC01 | Missing parameter.                              | ☾ |
| SC02 | Illegal parameter.                              |   |

- SC03            The specified program cannot be scheduled.
- SC04            The specified program is not a subordinate (or "SON") to the program issuing the completion call.
- SC05            The program given is not defined.  
                  The format for the SC05 error is:  
                  PROG/SEGMENT nnnnn SC05 PROG  
                  ADDRESS  
                  Where:  
                  nnnnn = program/segment not found.
- SC06            No resolution code is specified in the execution time EXEC call.
- SC07            A prohibited core lock was attempted.
- SC08            The program just scheduled is assigned to a partition smaller than the program itself or to an undefined partition.
- SC09            The program just scheduled is too large for any partition of the same type.
- SC10            There is not enough system available memory for the string passage.
- SC11            EXEC schedule or timed execution request was issued and program specified is already in the time list for another session.
- SC12            The program tried to do an EXEC 8 to load an MLS program.
- SC13            The main program and segments were not SP'ed onto the same disc cartridge.
- SC14            Track ownership in track assignment table does not correspond to ID segment for program's segment.

## ERROR CODES







### SCOM ERROR MESSAGES

ERROR MESSAGES	MEANING
WARNING: RECORD TOO LONG	A record has been read whose length is greater than the specified maximum record length.
FMP ERROR= -xxx ON FILE yyyyyy	An FMP error was encountered on the specified file.
USER SUPPLIED MAXIMUM RECORD SIZE IS TOO LARGE	There is not sufficient buffer space to accommodate the specified maximum record length.
ILLEGAL INTERNAL SUBROUTINE PARAMETER	Subroutine was called with invalid parameter.
CACHE DATA STRUCTURE CORRUPT	Internal check of buffer data structure shows corruption.



## ERROR CODES

### SYSTEM AND BREAK-MODE COMMAND ERROR MESSAGES

ERROR MESSAGE	MEANING	
OP CODE ERROR	Illegal operator request code.	
NO SUCH PROG	The name entered is not a main program in the system.	
INPUT ERROR	A parameter is illegal.	
ILLEGAL STATUS	Program is already scheduled.	
CMD IGNORED — NO MEM	Not enough system available memory exists for storing the program's command string	
ILLEGAL PART'N	Partition does not match command request.	
SIZE ERROR	Illegal program size specified or size of program specified larger than its assigned partition or any partition.	
xxxxx NO SWAP TRACKS	Not enough swap tracks available to swap out a program on behalf of program xxxxx.	
		

**SYSTEM BOOT-UP HALTS (front panel)**

HLT	MEANING
2	Memory wrap-around halt. Located in location 2 of the system map.
3	Memory wrap-around halt. Located in location 3 of the system map.
4	Powerfail occurred and powerfail automatic restart is enabled.
5	Memory protect switch was set and memory parity error occurred.
6	A partition was found that is not properly linked into an operating system partition list. The operating system may be corrupt.
10B	FMGR or D.RTR cannot be scheduled at startup because there is not a large enough partition (issued by the system).
11B	Attempt was made to re-execute a non-RPL compatible ROM Loader Part # 12992A, or Bootstrap Loader.
20B	Tried to find memory that does not exist.
21B	Bad VMA/OS firmware. This halt may appear as 105355B in the T- register.
22B	\$CNFG cannot find an ID segment for Configurator extension \$CNFX, \$CNFX is not a Type 3 program, or a contiguous memory block of three good pages cannot be found in the user partition area.
30B	Error was encountered in the disc I/O process by one of the RPL-compatible ROM Loaders Part # 12992B and 12992F. If the disc is a 7900 the disc



## ERROR CODES

status is displayed in the A-register. If the disc is a 7905/20 the disc status word 1 is displayed in the B-register and disc status word 2 in the A-register.

HLT

MEANING

31B

Error encountered in the disc I/O process by the Boot Extension. If the disc is a 7900, the disc status is displayed in the A-register. If the disc is 7905/06(H)/20(H)25(H), the disc status word 1 is displayed in the B-register and disc status word 2 is displayed in the A-register.

55B

An EQT with the equipment type code of console cannot be found.

56B,57B

While dispatching a program, the operating system encountered an unexplainable condition. The operating system may be corrupt.

## TRACK (DISC PARITY) ERROR MESSAGE

TR nnnnn EQT xx, U yy or  
S  
U

nnnnn = track number of track containing error

xx = EQT of disc

yy = subchannel of disc

S = system request encountered error

U = user request encountered error

## VMA/EMA ERROR CODES

VMA/EMA errors that cause a program to abort have the same format as the MP and DM error returns:

**VM xx**

or

**EM xx**

where

**xx** is an FMP error number (if xx is less than 80). FMP reports the error as a negative number and VMA/EMA reports the same error as the positive of that number. VMA/EMA errors with xx greater than 80 are not FMP errors.

- 01 Disc error.
- 02 Duplicate file name.
- 05 File extent cannot be created when read only access has been specified to the VMA file. X-reg = the requested page ID that caused the problem.
- 06 File not found.
- 07 Illegal security code or illegal write on LU 2 or 3.
- 08 File open or lock rejected.
- 12 File extent cannot be created when read-only access has been specified to the VMA file. X-reg = requested page that caused problem.
- 13 Specified cartridge is locked.
- 14 Directory full.
- 15 Illegal file name.
- 19 Illegal access on a system disc.
- 20 An array is specified with incorrect subscripts.
- 21 MSEG in the \$EMA directive is not specified correctly.
- 22 The program is not an EMA/VMA program.
- 32 Cartridge not found.
- 33 Not enough room on cartridge.
- 46 Greater than 255 file extents on the VMA file.
- 80 EMA/VMA system is corrupt.
- 81 Not an EMA/VMA program, or a bad request to VMAIO or XLUEX.

## ERROR CODES

- 82 Requested page beyond maximum page specified for EMA/VMA system or the VMA disc file is too small.  
X-reg = requested page number (in octal).  
Y-reg = logical address to map in the requested page.  
Abort address = address of instruction causing program to abort.
- 83 All pages locked; working set is not large enough to support the size of MSEG specified in your program.
- 84 The backing store file is not the correct type (file type 2) or the record length is not 1024 words.
- 85 Scratch file cannot be purged; file is in use by another program.
- 86 Access to VMA system after the VMA file has been closed.
- 87 MSEG is too small.
- 88 Cannot re-specify the VMA file.
- 89 Transfer too big for VMAIO or XLUEx.
- 90 Shareable EMA size for program is larger than the shareable EMA area already allocated.
- 91 Program and shareable EXM area are assigned to a reserved partition in which program's shareable EMA area has already been allocated.

## CI ERROR MESSAGES







CI error messages are self-explanatory. For further information, refer to CI User's Manual.

## FMP ERROR MESSAGES

ERROR	ERROR MESSAGE
-000	(no error)
-001	DISC ERROR
-002	FILE ALREADY EXISTS
-003	BACKSPACE ILLEGAL
-004	ILLEGAL RECORD LENGTH
-005	BAD RECORD LENGTH
-006	NO SUCH FILE
-007	BAD FILE SECURITY CODE
-008	FILE IS ALREADY OPEN

**ERROR**

**ERROR MESSAGE**

 -009	ATTEMPT TO POSITION OR FORCE TO 1 A TYPE 0 FILE
-010	NOT ENOUGH PARAMETERS
-011	DCB NOT OPEN
-012	ILLEGAL FILE POSITION
 -013	DISC LOCKED
-014	DIRECTORY IS FULL
-015	ILLEGAL NAME
-016	ILLEGAL TYPE OR SIZE=0
-017	ILLEGAL READ/WRITE ON TYPE 0 FILE
 -018	ILLEGAL LU. LU NOT ASSIGNED TO SYSTEM
-032	NO SUCH CARTRIDGE
-033	RAN OUT OF DISC SPACE
-036	LOCK ERROR ON DEVICE
-037	PROGRAM IS ACTIVE
-038	ILLEGAL SCRATCH FILE NUMBER
-046	GREATER THAN 255 EXTENTS
-049	COPY VERIFY FAILED
-050	NO FILES FOUND
-099	D.RTR REQUEST ABORTED
-101	ILLEGAL PARAMETER IN D.RTR CALL
-102	D.RTR NOT AVAILABLE
 -103	DIRECTORY IS CORRUPT
-104	EXTENT NOT FOUND
-200	NO WORKING DIRECTORY
-201	DIRECTORY NOT EMPTY
 -202	DID NOT ASK TO READ
-203	DID NOT ASK TO WRITE
-204	FILE READ PROTECTED
-205	FILE WRITE PROTECTED
-206	DIRECTORY READ PROTECTED
 -207	DIRECTORY WRITE PROTECTED
-208	DIRECTORY ALREADY EXISTS
-209	NO SUCH DIRECTORY

## ERROR CODES

ERROR	ERROR MESSAGE
-210	UNPURGE FAILED
-211	DIRECTORIES ARE NOT ON THE SAME LU
-212	CANNOT CHANGE THAT ATTRIBUTE FILE TYPE, SIZE, OR RECORD LENGTH.
-213	TOO MANY OPEN FILES
-214	DISC NOT MOUNTED
-215	TOO MANY DIRECTORIES
-216	YOU DO NOT OWN
-217	BAD DIRECTORY LOCK
-218	MUST SPECIFY AN LU
-219	NO REMOTE ACCESS
-220	DSRTR NOT AVAILABLE
-221	FILES ARE OPEN ON LU
-222	LU HAS OLD DIRECTORY
-223	ILLEGAL DCB BUFFER SIZE
-224	NO FREE ID SEGMENTS
-225	PROGRAM BUSY
-226	PROGRAM WAS ABORTED
-227	PROGRAM DOESN'T FIT IN PARTITION (SC08/09)
-228	NO SAM TO PASS STRING (SC10)
-229	ACTIVE WORKING DIRECTORY
-230	ILLEGAL USE OF DIRECTORY
-231	STRING IS TOO LONG
-232	UNKNOWN FOR OLD FILE
-233	NO SUCH USER
-234	SIZE MISMATCH ON COPY
-235	BREAKFLAG DETECTED
-236	RESERVED FOR SUPERUSER
-237	MUST NOT BE REMOTE
-238	ILLEGAL PROGRAM FILE
-239	PROGRAM NAME EXISTS
-242	DISC I/O FAILED
-243	PARAMETER ERROR
-244	MAPPING ERROR

<b>ERROR</b>	<b>ERROR MESSAGE</b>
-246	SYSTEM COMMON CHANGED
-300	ILLEGAL REMOTE ACCESS
-301	TOO MANY REMOTE CONNECTIONS
-302	NO SUCH NODE
-303	SESSION LIMIT EXCEEDED
-304	NO SUCH ACCOUNT
-305	BAD PASSWORD
-306	CAN'T ACCESS ACCOUNT
-308	CONNECTION BROKEN
-310	DS IS NOT INITIALIZED
-311	DS LINK IS NOT CONNECTED
-312	REMOTE SYSTEM DOESN'T RESPOND
-313	NO TRFAS AT REMOTE SYSTEM
-315	DS ERROR DSXX(X), NODE YY

