HEWLETT PACKARD

Computer Museum

RTE-A
Quick Reference Guide

2

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed that contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine which manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

```
First Edition ............... May 1982 ...........................
    Update 1 ............. Jul  1982 ...........................
    Update 2 ............. Jan  1983 ...........................
Reprint .................... Jan  1983 Updates 1 and 2 incorporated
Second Edition ............ Jun  1983 ...........................
    Update 1 ............. Dec  1983 ...........................
Reprint .................... Dec  1983 ...... Update 1 incorporated
Third Edition .............. Jan  1985 ...........................
    Update 1 ............. Dec  1985 ...........................
Fourth Edition ............. Aug  1986 ...........................
Fifth Edition .............. Feb  1988 Software Rev. 5.0 (Rev. 5000)
Sixth Edition .............. Jan  1989 Software Rev. 5.1 (Rev. 5010)
    Update 1 ............. Oct  1990 Software Rev. 5.2 (Rev. 5020)
Seventh Edition ........... Dec  1992 Software Rev. 6.0 (Rev. 6000)
Eighth Edition ............. Nov  1993 Software Rev. 6.1 (Rev. 6100)
```

Conventions Used In This Manual

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Command Syntax

This manual uses the following conventions to describe command syntax:

CAPITAL LETTERS 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.

| Parameter choices are separated by vertical bars.

, Delimiters between commands and parameters are commas or spaces.

lowercase italics Terms that must be replaced by actual parameters (variables) are in lowercase italic letters.

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

prog 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.)

lu Logical unit number, in the range 0 to 255, inclusive. Refers to a physical I/O device. LU 1 refers to the user terminal. LU 0 refers to "the bit bucket", a nonexistent device (unwanted data can be sent to LU 0).

file (or *filedescriptor*), unambiguously specifies a single file.

directory A directory or subdirectory.

file|*lu* Either a file descriptor or an LU can be specified.

mask|*lu* Either a mask or an LU can be specified.

mask Mask field, a file descriptor that can include the two "wild card" characters, dash (−) and at-sign (@), and a mask qualifier. Mask qualifiers are:

(*user.group*)	match *user.group* name
a	access date range
b	backup files
c	creation date range

d	match any directory
e	search every disk volume
k	search from the specified directory down
m	return extent entries on FMGR directories
n	not(d)
o	open files
p	purged files
s	search all subdirectories
t	match only temporary files
u	update date range
w	walk (do not run) through FMGR directories
x	files with extents

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

. a [*yymmdd* . *hhmmss*] – [*yymmdd* . *hhmmss*]

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

pram   One parameter is allowed.

pram*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.

Command Stack Editor

Purpose: Allow previously entered command lines to be displayed, edited, and re-entered.

Syntax: / [:] [/ . . / | n] [. *pattern*]

: denotes auto-execute mode.

/ . . / number of slashes specifies line number at which to start frame.

n line number at which to start frame.

. *pattern* select lines containing this pattern; pattern syntax is:

[^] { - | @ | *char* | \ *char* } . . .

  ^ anchors pattern to start of command line.
  - matches any single character.
  @ matches zero or more characters.
  char matches specified character.
  \ *char* matches specified character, allowing − or @ to be entered.
  \ quoting character.

The command stack editing mode commands are:

  ctrl-A Go to start of line where cursor is positioned.
  ctrl-Z Go to end of selected line.
  ctrl-P Display previous frame of selected lines.
  ctrl-F Display following frame of selected lines.
  ctrl-K Mark current line for grouped execution in order of marking.
  ctrl-D Delete current line from stack.
  ctrl-Q Quit stack mode, start executing marked lines.
  ctrl-U Enter instead of *ctrl-Q* on terminals using Xon/Xoff handshake protocol.

ctrl-Q ctrl-Q, ctrl-U ctrl-U
  Abandon stack mode, forget marked lines.

System> and RTE: Prompts

Commands marked with an asterisk (*) can be used in response to the System> and RTE: prompts, as well as the CI> prompt.

File Descriptors

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

Standard 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 file name has up to 16 characters, plus a file type extension, and should not include the characters at-sign (@), minus (−), greater than (>), comma (,), slash (/), period (.), or left bracket ([).

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

.c	C source file
.cmd	command file
.dat	data file
.dbg	debug file
.dir	directory of subdirectory entry
.doc	document file
.err	eror message file
.ftn	FORTRAN source file
.ftni	fortran source include file
.h	C include file
.hlp	help file
.lib	indexed library of relocatables
.lod	LINK command file
.lst	listing
.mac	Macro source file
.maci	Macro source include file
.map	loader map listing
.merg	merge file for relocatables without headers

.mlb	Macro library file.mrg library merge file for relocatables with headers
.mnf	manual numbering file
.pas	Pascal source file
.pasi	Pascal source include file
.rel	relocatable (binary) file
.run	program file
.snp	system snapshot file
.spl	spooling system file
.stk	command stack file
.sys	system file
.txt	text file

File type is specified by an integer in the range -1 to 32767. Default is type three. Standard file types are:

-1	Symbolic link. (VC+ only)
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 Procedure

From the Virtual Control Panel (VCP) terminal:

1. If not in VCP mode, press the BREAK key to pass control to
    VCP. Now VCP commands can be used to boot the system.

2. Enter %B*dvfffffbusctext*

    where:

    dv   is one of the following device types:

    DC   Disk (cartridge or flexible) via HP-IB or SCSI
    DI   Disk interface
    MT   Magnetic tape
    DS   DS computer network link or LAN
    RM   PROM card

    The default parameters are:

    For DC: %BDC002027BOOT.CMD for HP-IB
    For DC: %BDC006027BOOT.CMD for SCSI
    For DI: %BDI000032BOOT.CMD
    For DS: %BDS000024
    For RM: %BRM000022

    fffff   File number (octal 0 to 77777 only).

    b   4k-word memory block number when reading from
                cartridge tape; HP-IB or SCSI bus address when read-
                ing from disk drive; otherwise, zero.

    u   Unit number (zero to seven) only if used on device.
                For a CS/80 disk that includes a cartridge tape drive
                (CTD), unit 0 is the disk drive and unit 1 is the CTD.

    sc   Select code of the interface card to be used.

    text   File name, or ASCII string to be passed to the pro-
                gram after it is loaded.

    Leading zeros can be omitted.

CI Commands

ALIAS (Define/Display Aliases; VC+ Only)

Purpose: Defines a CI alias or displays all previously defined aliases.

Syntax: ALIAS [-x|+x] [*alias_name* [[=] *string_value*]]

ALIAS *alias_name*

-x   If the user session has an EVB, this option exports a defined alias or defines a new one as exported. Exported aliases are available to all copies of CI in the session, including CM.

+x   This option imports an exported alias or defines a new alias as imported. An imported alias is defined only for the current CI.

If neither the −x or +x option is given, the alias is defined as a local alias.

alias_name is a string of up to 32 letters, digits, and underscores, not starting with a digit.

string_value
is a string terminated by the end of the command line or a ";", whichever comes first. Command lines can be 255 characters.

AS (Assign Partition)*

Purpose: Assign a program to a reserved partition.

Syntax: AS *prog partNum* [C|D]

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

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

C (CODE) assign code (executable) section of the program to the reserved partition.

D (DATA) assign the data section of the program to the reserved partition.

Default is the data section (D).

---

* This command can also be entered in response to the System> or RTE: prompt.

ASK (Display a Prompt and Read a Response)

Purpose: Display a question or prompt, read response from the terminal and pass back to scheduling program in returns.

Syntax: ASK *'character string'*

character string

Any question or prompt the user desires. The string must be enclosed in backquotes (' ').

AT (Set Program Runtime)

Purpose: Set the execution time of a program, or set a program to run at regular intervals.

Syntax: AT *time* [*intvl*] *prog|file* [*pram*5]

time   Program runtime in 24-hour or 12-hour format (program is scheduled immediately if time = 0):

hr:min:sec   (for example, 13:30 for one thirty pm)
        or
hr:min:sec AM|PM   (for example, 1:30 pm)

intvl   Optional rescheduling interval (0 to 4095) specified with one of the following units:

MIL   for milliseconds
S or SEC   for seconds
M or MIN   for minutes
H or HOUR   for hours

The start time must always be specified.

prog|file   A 5-character program name or a file descriptor that identifies a program (type 6) file.

*pram*5   Parameters passed to program. Up to five parameters are allowed and up to 256 characters for the runstring.

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.

CD (Change Working Directory)

Purpose: Changes the working directory.

Syntax: CD [- | *directory*]
              CD *old new*
              CD [-p] [*directory*] (VC+ Only)

   -         Change current directory to the previous directory
              ($OLDPWD).

  directory   Change current directory to *directory*. The default for
              directory is the value of the $HOME variable.

  old new   Substitute the string *new* for the string *old* in the
              current working directory name, $WD, and change to
              this new directory.

   -p   When using a ksh-style editing mode, the commmand
              "cd .." moves the current directory one path
              component closer to the root directory. This option
              preserves the physical path when treating symbolic
              links. "cd -p .." changes the working directory to
              the parent directory of the current working directory.
              This option is only available for the ksh-style editing
              modes.

CL (List Mounted Disks)

Purpose: Display all mounted disk volumes.

Syntax: CL

---

* This command can also be entered in response to the System> or RTE: prompt.

CN (Control Device)

Purpose: Control peripheral devices.

Syntax: CN *lu function*   [*pram*4*]

lu   Logical unit of device to receive the control request.

function   The control function code (0-63B) as defined in the
            function field of the control word (listed for each driver
            in the *RTE-A Driver Reference Manual*). A 2-character
            mnemonic code can be used for some of the more
            commonly used control functions. The action
            performed for a particular function code is dependent
            on the driver. Refer to the *RTE-A Driver Reference
            Manual* for a complete list of the function codes
            available for a particular driver.

Mnemonic Code	Octal Code	prams Required	Action
AB	None	None	Abort current request (at head of queue)
AD	24B	Address	Establish new device address
TO	11B	# lines on page	Issue top-of-form or line spacing on printer
RW	4	None	Rewind cassette tape
EO	1	None	Write end-of-file
FF	13B	None	Forward space file
BF	14B	None	Backward space file
FR	3	None	Forward space record
BR	2	None	Backward space record
DP	25B	1-4 prams	Set device prams
None	0	None	Clear device
None	20B	Prog	Enable primary program
None	21B	Prog	Disable primary program
None	40B	Prog	Enable secondary program
None	41B	Prog	Disable secondary program

pram*4   Optional parameters. These parameters are described
            in the *RTE-A Driver Reference Manual* for each driver.

            If a parameter is an ASCII character string, it is parsed
            into two-byte words, which are passed to the driver as
            separate parameters.

CO (Copy Files)

Purpose:	Copy one or more files between directories and/or I/O devices.
Syntax:	CO *mask* | *lu1* *mask* | *lu2* [*pram*]
mask | *lu1*	Source file or device
mask | *lu2*	Destination file or output device.
pram	One of the following characters (default is A):

A	ASCII records; no checksum.
B	Binary absolute; checksum performed.
C	Clear backup bit on source after copying.
D	Overwrite duplicate files.
N	No carriage control on source.
P	Purge source after copying.
Q	Do not record access time on source.
S	Preserve directory information (timestamps, protection, and backup bit) of the source file.
T	Truncate destination to length of valid data.
U	Replace duplicates if update time is older.

CP (Copy Files; VC+ Only)

Purpose:	Copies files and directory subtrees.

Syntax:

```
cp [-F|-I] [-PQRV] file1 dest_file
cp [-F|-I] [-PQRV] file1|mask1
                        [file2|mask2...] dest_dir/
cp [-F|-I] [-PQRV] file1|mask1
                        [file2|mask2...] dest_mask
cp [-F|-I] [-PQV] -R dir1/ [dir2/...]
                                     dest_dir/
```

-F	Forced copy - purge existing destination file before each copy without prompting for confirmation. Only write access to the directory is required.
-I	Interactive copy - issue a prompt requesting confirmation for each copy that would overwrite an existing file. The −I option is ignored if the −F option is also set.
-P	Preserve - preserve the directory attributes when copying files; ownership of directories is also preserved.

-Q	Quiet - inhibit error/warning reporting.
-R	Recursive copy - recursively copy a directory subtree to another directory. When the destination directory already exists, the source directory and all of the files under it are copied to the destination directory. When the destination directory does not exist, it is created and all of the files under the source directory are copied to the destination directory.
	When the –R option is specified, symbolic links are copied such that the target points to the same location as the source.
-V	Verbose mode.
--	Indicates the end of the options (required if *mask* begins with '–').

file1 ...
mask1 ...
dir1 / ... One or more files or directories to be copied.

dest_dir /
dest_mask The directory or destination file mask to which the file or directory is to be copied. When files are copied to *dest_dir*, the files are copied into the directory. When a source mask is specified (*mask1 ...*) or when two or more files are to be copied, the destination must be a directory or a destination mask.

When specifying directories in the argument list, a directory may optionally be specified with a trailing slash (*dir/*) or with the type extension (*dir*.DIR). The trailing slash or the type extension is necessary when a file name without a type extension exists that has the same name as the directory.

CR (Create File)

Purpose: Create a disk file.

Syntax: CR *file*

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

STANDARD:
  /dir/subdir/filename : : : type : size : recLength

COMBINED:
  subdir/filename : : dir : type : size : recLength

FMGR:
  filename : sc : crn : type : size : recLength

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 disk, typically 48 or 64 blocks for a hard disk and 30 or 16 for a flexible disk. Directory size is extended as needed.

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

CZ (Display/Modify Code Partition Size; VC+ Only)*

Purpose:	Display or modify the code partition size of a CDS program.
Syntax:	CZ *prog* [*segNum* | AL]
prog	Program name.
segNum	The number of segments to be included in the calculation of partition size. This number must be less than or equal to the actual number of segments.
AL	Include all segments in calculation of partition size.

DC (Dismount Disk Volume)

Purpose:	Dismount a disk volume.
Syntax:	DC *lu*
lu	LU number of the disk volume to be dismounted. Can be positive or negative.

---

* This command can also be entered in response to the System> or RTE: prompt.

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.
E	Type EXTENSION (for sorting only).
F	FILE type.
L	File LOCATION (block address and LU on disk).
M	MAIN file size in blocks, excluding extents.
N	NUMBER of records.
O	Display OPEN files.
P	File PROTECTION in the form *owner[/group]/other*.
R	Length (in words) of the 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).
Z	Display contents of symbolic links.
*	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 ! option is specified with a FMGR file.

DT (Display/Modify Data Partition Size; VC+ Only)*

Purpose: Display or modify the data partition size of a CDS program.

Syntax: DT *prog* [*size* [*emaSize* [*msegSize*]]]

    prog   Program name

    size   Size of the data partition, in pages.

    emaSize   Number of pages needed for EMA.

    msegSize   Size of data space mapped onto the EMA.

Note: Data partition size + *msegSize* must be < 32 pages.

ECHO (Display Parameters to Terminal)

Purpose: Display parameters, separated by commas, at the terminal.

Syntax: ECHO [*parameters*]

parameters   One or more parameters separated by blanks or commas. Positional, user-defined, and predefined variables can be included in the string. If this parameter is omitted, a blank line is displayed.

EX (Exit)

Purpose: Terminate CI and print the message "Finished".

Syntax: EX [*option*]

    option   Specifies action if programs are active:

      B   Make background session (non-interactive, multi-user environment only).

      C   Continue execution; do not log off.

      L   Log off; abort active programs.

---

* This command can also be entered in response to the System> or RTE: prompt.

FUNCTION (Define a Function; VC+ Only)

Purpose: Defines a CI function.

Syntax: FUNCTION [-x|+x] [*function* {
        cmd_line1
        cmd_line2
        ...
        cmd_lineN
        }]

-x   Export a defined function or define a new one as exported. An exported function is available to all copies of CI in the session.

+x   Import an exported function or define a new function as imported. An imported function is defined only for the current CI.

           If neither the −x or +x option is given, the function is defined as a local function.

function   A string of up to 32 letters, digits, and underscores, not starting with a digit.

cmd_line1 CI command lines.
...
cmd_lineN

FUNCTIONS (Display Functions; VC+ Only)

Purpose: Displays all or specific defined functions.

Syntax: FUNCTIONS [+x|-x|*function*]

+x   Display only local functions.

-x   Display only exported functions. This option is only applicable if the session has an EVB.

function   A string of up to 32 letters, digits, and underscores, not starting with a digit.

GO (Resume Suspended Program)*

Purpose: Resume execution of a suspended program.

Syntax: GO [*prog* [*pram**5]]

   prog   Name of the suspended program.

  pram*5   Parameters to be passed to the program only if the
             program has suspended itself.

GREP, FGREP (Search a File for a Pattern)

Purpose: Searches a file for a pattern. These commands support
             redirection.

Syntax: grep [*options*] [*expression*] *file* | *mask* . . .

          fgrep [*options*] [*string*] *file* | *mask* . . .

  options   One or more options listed below. Options that do not
           require an argument may be grouped together after a
           single hyphen and may be in any order. To specify an
           option that requires an argument, the option must
           appear either by itself (for example, −e *expr*) or at the
           end of a group of options (for example, −nbe *expr*).

        −b   Each line is preceded by the block number on
            which it was found. Block numbers are calculated
            as follows: (the number of bytes that have been
            read from the file)/256; the result is rounded
            down.

        −c   Only a count of the lines containing a match is
            printed.

        −d   Directory display mode. Displays directory names
            on separate lines.

       −e *expr*
            Same as the expression argument in the runstring.
            Must be used when the expression begins with a
            hyphen (−). May also be used multiple times to
            specify more than one expression for which to
            search.

---

* This command can also be entered in response to the System> or RTE: prompt.

−g /*expr*/
Same as the −e option, but the expression is delimited by slashes. Useful when the expression contains a hyphen (−) or a comma (,).

−h   Display help information for GREP or FGREP.

−l   Only the names of files that contain a match are listed. File names are listed once, each separated by a new line.

−n   Each line that contains a match is preceded by its relative line number in the file starting at line 1.

−r   Right justify block numbers (−b) and/or line numbers (−n).

−s   Supress error messages produced for nonexistent or unreadable files.

−t   Search only text (types 3 and 4) files.

−u   Case sensitive match. Note that you must protect the search pattern from CI, see below.

−v   All lines that do not contain a match are printed.

−x   Exact match; only lines matched in their entirety are printed (FGREP only).

−z   Display the file name of each file being searched.

−−   Marks the end of the options.

expression   The regular expression pattern (GREP) or string
string   (FGREP) to be used for the search pattern. Spaces and upper and lowercase may be protected from CI by enclosing the pattern in backquotes (`), for example `This Pattern`. If backquotes are not used, the string is upshifted and commas are inserted as delimiters. For information on regular expressions, see the *EDIT/1000 User's Manual*, part number 92074-90001..

file | mask   The file(s) to be searched; up to 10 file names or masks may be specified.

IF-THEN-ELSE-FI (Control Structure)

Purpose: Allow decision making in a command file.

Syntax: IF *cmd-list1*
              THEN *cmd-list2*
              [ELSE *cmd-list3*]
              FI

cmd-listn   A list of commands either one command per line or
              multiple commands per line separated by semicolons.
              A command list can be null.

IN (Initialize Disk Volume)

Purpose: Prepare a blank disk volume for system use.

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

lu   LU number of the disk volume to be initialized.

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

OK   Suppresses user prompt.

IO (Display I/O Configuration and Status)

Purpose: Display the system I/O configuration.

Syntax: IO [*-flags*] [*args*] [*listfile*]

flags   A string of single alphabetic characters preceded by a
              dash; for example, −CX selects both the C and X flags.
              flags select the style of report IO displays. Valid flags
              are:

              C   Show configuration, not status, information. If
                   this flag is not present, device status information
                   is displayed.

              D   For the device status report, D inhibits looking up
                   device descriptions in file /SYSTEM/DEVICES,
                   using the generic device type descriptions instead.

L   Sort the select code configuration listing by LU within the select code.

S   Organize the configuration report by select code, not LU.

X   Display extended configuration information for each device.

args   Takes one of three forms, depending on the flags selected. If the S flag is not selected, it is of the form:

firstlu [*lastlu*]
or
symlink_file

where:

firstlu   is the first LU to be displayed.

lastlu   is the last LU to be displayed. If *lastlu* is not given, only *firstlu* is displayed. If neither *firstlu* nor *lastlu* is given, all LUs are shown.

symlink_file
is the name of a symbolic link file that points to an LU. Information for that LU will be displayed.

LUs may be given as an octal value suffixed with "B".

If the S flag is selected, *args* is of the form:

sc   the select code of the interface card to display. This may be given as an octal value suffixed with "B". If not given, all select codes are shown.

listfile   Name of the file to which the listing is to be sent; if none is specified, the listing is displayed at the scheduling terminal.

IS (Compare Strings or Numbers)

Purpose: Compare two character strings or numbers.

Syntax: IS *string1 rel_oper string2* [*option*]

string1   A numeric or character string

rel_oper   Relational operator indicating the relation being tested. The two sets of operators recognized are as follows:

=	or EQ	Equal to
<>	or NE	Not equal to
<	or LT	Less than
<=	or LE	Less than or equal to
>	or GT	Greater than
>=	or GE	Greater than or equal to

string2   A numeric or character string.

option   Specifies special comparison instructions. The possible values are as follows:

-I   Integer comparison. A suffix of B following *string1* or *string2* in either upper or lowercase indicates an octal value. A leading minus (−) is accepted for decimal values.

-A   Do not fold alphabetic characters before comparison.

LI (List Files)

Purpose:	List files to terminal.
Syntax:	LI [-*flags*] *filemask*
filemask	File descriptor mask for files to be listed.
flags	Precede flags with a dash. Group them together, as in "-nhx", or separately, as in "-n -h -x"; "-s 10 -e 15" is equivalent to "-se 10 15".

-a	List ASCII text (default for file types 0, 3, and 4).
-w	List octal words (default for file types other than file type 0, 3, and 4).
-o	List octal bytes.
-i	List signed integer words.
-b	List binary words.
-h	List hexadecimal words.
-d	List ASCII text with Display Functions around special characters.
-n	List line/record numbers.
-s *ln*	Set listing to start at *ln*.
-e *ln*	Set listing to end at *ln*.
-x	Quit listing at EOF and do not prompt.
-$	Always prompt at the end of file, even if the file is less than one page long.
-m	Fold long lines; treat lines longer than 79 characters as multiple lines for pagination.
-t	Truncate trailing blanks on text listings.
-f	Force type-1 access, list blocks in octal words.
-c	File has carriage-control characters in column 1.
-q	Quiet file access, do not record access time.
-l *fl*	Divert listing to file *fl*. Precede *fl* with tilde (~) to overlay existing file or with plus (+) to append to existing file.
-y	Yes, list each file matching *filemask*.
-r *rsz*	Set maximum record size if more than 512 characters desired.
-*pgsz*	Set lines per page to *pgsz* (1..32767); do not paginate if *pgsz* is zero.

-> /*cmds*/

> Execute initial command string *cmds* at start of file.
>
> Use back quotes (' ') around string entered from CI.

-p /*str*/

> Redefine 'More...' prompt. Delimit *str* with character other than space or comma. Use back quotes (' ') around string.
>
> The following substitutions occur within *str*:
>
> | | |
> |---|---|
> | %f | file name. |
> | %l | current line number. |
> | %p | percentage through file as in default. |
> | %w | window or page number viewed. |
> | %% | %. |

The following listing commands can be entered at the 'More...' or 'End...' prompts; *n* is a number (from 1 to 2147483647) that can precede the listing command:

space or 1   List next page or next *n* lines.

return   List rest of file or goto line *n*.

a or q   Abort list; 'a' moves to next masked file, 'q' quits entire listing.

+   List next line or skip forward *n* lines.

–   Skip backward 1 line or *n* lines from top line in window.

b   Skip backward 1 page or *n* pages from top line in window.

g or .   Goto line *n*; 'g' to list page, '.' to list 1 line.

$ List last window.

%   Goto line *n* percent through file.

'*rex*'   Search for next occurance of regular expression, *rex* from current window or line *n*. Null string searches for the last string entered. In interactive mode, eliminate trailing apostrophe and terminate with <return>.

f /*rex*/   Same as '*rex*' except with user-defined delimiters. In interactive mode, use <return> instead of delimiters.

`rex`	Search backward for regular expression *rex* from current window or line *n*. Null string searches for last string entered. In interactive mode, eliminate trailing backquote and terminate with <return>.
@ /*rex* /	Show all lines containing pattern from the current window or from line *n*.
k*m*	Mark top window line or line *n* with character *m*, which must be an alphabetical character ('A'..'Z').
:*m*	Go to line marked with character *m*; list *n* lines.
u*m*	List until line marked with *m*; list no more than *n* lines.
p	Set page size to *n* and list a page.
Oc	Toggle or reset runstring flags, including listing modes (a, b, h, i, o, or w).
s	Set starting file line to window top. LI never starts before this line.
e	Set ending file line to window bottom. LI never advances past this line.
#f	Move to file #*n* or prompt for file number *f*. (Requires VMA.)
#+[*i*]	Move forward *i* selected files or 1 file. (Requires VMA.)
#-[*i*]	Move backward *i* selected files or 1 file. (Requires VMA.)
#?[*f*]	Show window of selected files around current file or of files starting at *f*. (Requires VMA.)
=	Display file name, current line.
n*file*	Add new file name, *file*, to list of files to be displayed and move to this file. (Requires VMA.)
r	Purge file listed.
? or h	Help.
z	Suspend LI; restart with system GO command.

When more than one file is selected, LI prompts with

```
File:   file, list? [Y]
```

before listing each file (the "y" option turns this prompt off). When
prompted, you may enter one of the responses shown below:

Y or <space> or <return>
:   List the named file.

N
:   Do not list the file.

S
:   List the file and set the "y" option (to suppress the prompt).

A
:   Do not list the file; abort the listing.

#
:   Show selected files or move to another file (see description of the # command above).

R
:   Remove (purge) this file.

Regular expressions are the same as those in Edit/1000; see the
documentation for Edit/1000 for use and examples. In brief, the
expressions are shown below:

.
:   Matches any character.

@
:   Matches any character zero or more times (same as ".*").

^x
:   Anchors the pattern to the beginning of the line.

x$
:   Anchors the pattern to the end of the line.

[ai-k]
:   Matches any of the characters "a", "i", "j", and "k".

[^ai-k]
:   Matches any character except "a", "i", "j", and "k".

x*
:   Matches zero or more occurrences of pattern x.

x+
:   Matches one or more occurrences of pattern x.

x<5>
:   Matches 5 repetitions of pattern x.

a:b
:   Matches a word boundary between patterns a and b.

*
:   Matches the character "*".

Backward movement is allowed for device (as opposed to disk) files,
but LI cannot back up beyond the number of lines that fit inside an
internal buffer.

LNS (Create Symbolic Link; VC+ Only)

Purpose: Create a symbolic link to a file or a directory.

Syntax: LNS [-FIQV] *file symlink_file*
                 LNS [-FIQV] *file1* | *mask1* [*file2* | *mask2...*] *dest_dir* /
                 LNS [-FIQV] *file1* | *mask1* [*file2* | *mask2...*] *dest_mask*
                 LNS [-FIQV] *dir symlink_dir*

-F   (Forced) overwrite existing files when creating symbolic links.

-I   (Interactive) issue a prompt requesting confirmation for each symbolic link that would overwrite an existing file. This option only has an effect when used with the −F option.

-Q   Quiet mode; inhibit error reporting.

-V   Verbose mode.

--   Specifies the end of options; required if *mask* begins with a hyphen (−).

file   File names, directory names, or masks that are to be
mask   used as the contents of the symbolic link files to be
dir   created.

symlink_file
            Name of the symbolic link file that is to be created.

dest_dir / If specified, LNS creates symbolic links in *dest_dir* to all of the files specified in the "*file1* | *mask1*, ..." arguments.

dest_mask If specified, LNS generates the corresponding destination file descriptor to create the symbolic link for each of the files specified in the "*file1* | *mask1*, ..." arguments.

LS, LL, LSF, LSX (List Directory Contents)

Purpose: Lists the contents of a directory. These commands support redirection.

Syntax:
$$\begin{bmatrix} \text{LS} \\ \text{LL} \\ \text{LSF} \\ \text{LSX} \end{bmatrix} \quad [-options] \quad mask1 \quad [mask2 \ \dots \ ]$$

-options   One or more of the following:

-C   use create time for sorting (−T) or listing.

-D   if an argument is a directory, list only its name; often used with −L to get status of a directory.

-G   same as −L, except only group is listed. If both −L and −G are specified, owner is not listed.

-L   list in long format; includes file attributes, owner, group, size in bytes, and the update time for each file. The field used by HP-UX ls command to specify number of links to a file is set to 1 on RTE-A. If time of last update is set to a date more than six months prior to the current date, or any time in the future, the year is substituted for the hour and minute of the update time. For remote files, owner's UID and group's GID numbers are displayed.

-N   use owner's UID and group's GID numbers rather than the associated character strings.

-O   same as −L, except that only owner is listed. If both −L and −O are specified, group is not listed.

-P   if a file to be listed is a directory, include a slash (/) after the file name.

-R   reverse the order of the sort. This produces a listing in descending order or oldest first (−T).

-T   sort according to the time modified (latest first); then sort alphabetically.

-U   use access time instead of update time when using the −T or −L option.

-X   output the listing in multiple column format; sort across instead of down the page.

-Y   display file type in the first character of the file attributes. Only RTE-A file types 1 through 9 are reported.

-z	display the file name as a namr. Ignored when used with multiple column output.	
-1	list file names in single column format regardless of the output device. This option overrides $-\backslash c$.	
$-\backslash c$	list output in multiple column format; sort down columns. The -1 option takes precedence over $-c$.	
$-\backslash f$	the following characters are included after each file name for the given types of files:	
	/	directory
	*	RTE type 6 file or CI command file
	@	symbolic link file
$-\backslash 1$	if the argument is a symbolic link, list directory attributes of the file or directory to which the link points rather than the link itself.	
--	marks the end of the options. Only required when the mask begins with a hyphen $(-)$.	
mask1 ...	One or more file masks for which to display file or directory information.	

MC (Mount Disk Volume)

Purpose: Mount a disk volume and make its contents available.

Syntax: MC *lu*

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

MO (Move Files)

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

Syntax: MO *mask1 mask2*

   mask1   Source file.

   mask2   Destination file.

MV (Move/Rename Files/Directories; VC+ Only)

Purpose: Moves or renames files and directories.

Syntax: mv [-F|-I] [-QV] *file1 dest_file*
     mv [-F|-I] [-QV] *file1|mask1* [*file2|mask2...*]
                    dest_dir/
     mv [-F|-I] [-QV] *file1|mask1* [*file2|mask2...*]
                   dest_mask
     mv [-F|-I] [-QV] *dir1/* [*dir2/* ...] *dest_dir/*

-F   Any existing destination file is purged before each move without prompting for confirmation. Only write access to the directory is required.

-I   Issue a prompt requesting confirmation for each move that would overwrite an existing file.

-Q   Quiet mode; inhibit error reporting.

-V   Verbose mode.

--   End of options; required if the mask begins with a hyphen (−).

file1 ... One or more files or directories to be moved.
mask1 ...
dir1/ ...

dest_dir/ Directory or destination file mask to which the file or
dest_mask directory is to be moved. When files are moved to *dest_dir*, files are moved into the directory. When a source mask is specified (*mask1 ...*) or when two or more files are to be moved, the destination must be a directory or a destination mask.

      When specifying directories in the argument list, a directory may optionally be specified with a trailing slash (*dirname/*) or with the type extension (*dirname*.DIR). The trailing slash or the type extension is necessary when a file name without a type extension exists that has the same name as the directory.

OF (Stop or Remove Program)*

Purpose: Terminate a program and, optionally, remove its program ID segment or remove a prototype ID segment.

Syntax: OF [*prog* [*opt*]]

    prog   Program name (up to 5 characters).

    opt   Either of the following:

        ID   release named program ID segment.

        D   release named prototype ID segment.

OWNER (Display/Change Owner; VC+ Only)

Purpose: Display or change the owner of a CI volume, a directory or a subdirectory.

Syntax: OWNER *directory* | *lu*V [*newOwner*]

    directory   Name of directory; no wild card characters allowed.

    *lu*V   A CI volume number followed by a 'V'.

    newOwner   Name of new owner; if omitted, the name of the owner is displayed.

---

* This command can also be entered in response to the System> or RTE: prompt.

PATH (Display/Change UDSP)

Purpose: Display or modify a User-Definable Directory Search
               Path (UDSP).

Syntax: PATH [-E]

               PATH [-E] [-N:n] *udsp* [*dir1* [*dir2*...[*dirN*]]]

               PATH [-E] -F *file* | *lu*

-E   Turn off echo; non-error messages are not displayed.

-N:*n*   Display or modify the specified entry. Set *n* equal to 1
               for UDSP #0 (home directory); otherwise, set *n* to a
               value between 1 and the UDSP depth.

udsp   Specifies the UDSP number. The values for *udsp* are
               as follows:

               0   Home directory.

               n   UDSP number between one and the number of
                     UDSPs defined for this session (maximum is 8).

               -A All UDSPs defined for current session.

dir   Specifies the directory name. The following special
               characters can be used:

               .   Use the working directory that is current when the
                   UDSP is referenced.

               !   Delete this UDSP or entry; this character must be
                   the only *dir* in the command line.

-F *file* | *lu*   Indicates that the commands will be input from the
                   specified file or LU.

POLL (Polling Function)

Purpose: Executes a specified CI command synchronously with respect to user interaction and terminal timeouts.

Syntax: POLL *intrvl* | OFF *command*

intrvl | OFF If a number, it is the approximate number of minutes between executions of the poll command.

            If OFF, the poll function is turned off.

command Any CI command/program to be executed at the poll interval.

PR (Display/Change 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.

PROT (Display/Change Protection; VC+ Only)

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

Syntax: PROT *mask* | *lu*V [*newProt*]

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

*lu*V   A CI volume number followed by a 'V'.

newProt   Access allowed owner, group and other users (r = read, w = write). If omitted, the current protection status is displayed. Syntax of *newProt* is:

             owner [/*group*] /*others*

---

* This command can also be entered in response to the System> or RTE: prompt.

PS (Display Program Status)*

Purpose: Display status of specified program or currently
executing program.

Syntax: PS [*prog*]

   prog   Name or ID segment of program for which status is
required.

If you omit the *prog* parameter, the status of the currently executing
program is displayed. If the specified program does not exist, the
following message is displayed:

```
No ID segment for this program. Try RP,program.
```

If the program does exist or if you enter the command without the
prog parameter, the response is displayed using the following format:

     prog PR(*priority*) PC(*addr*) [M] [T] [L] [S] *status*

where:

   prog   Program name.

   priority   Priority of the program.

   addr   Memory address of the last instruction executed.

   M   Program is memory-resident.

   T   Program is time scheduled.

   L   Program is being loaded from disk.

   S   Program is being swapped to disk.

The valid codes for the *status* parameter are as follows:

AB	The program is being aborted.
BL(*lu*)	The program has requested buffered output or Class I/O to the given LU, but the device already has requests exceeding its allotment of SAM.
CL	Either the program issued a Class I/O Get (EXEC 21) and the queue for the requested class number is empty, or the program attempted to allocate a class number and none was available.

---

* This command can also be entered in response to the System> or RTE: prompt.

DL(*lu*)　　The program has been suspended because it made an I/O request to the given LU and the LU is down.

IO(*lu*)　　The program is waiting for an I/O request to complete.

LD　　　　The program made an EXEC 28 or EXEC 8 request to restore a real-time program or statement from the disk. The program will continue when the load is initiated.

LK(*lu*)　　The program has been suspended because it made an I/O request to the given LU and the LU is locked to another program.

LK　　　　The program made an EXEC 28 or EXEC 8 request to restore a real-time program or statement from the disk. The program will continue when the load is initiated.

MM　　　　The program is waiting for SAM to become available. SAM is required for buffered output, Class I/O, and string passing.

OF　　　　The program is dormant. (The program either has never been scheduled, has run to completion, or has been aborted.)

PA　　　　The program has suspended itself with a PAUSE statement or an EXEC 7 call. You can reschedule the program with parameters (see the GO command).

QU(*progb*)　The program attempted to schedule *progb* with an EXEC 23 or 24 call, but *progb* was already active. When *progb* becomes dormant, it will be scheduled as the child of the program.

RN(*progb*)　The program is waiting for *progb* to release a resource number.

RN(-GLB)　The program is waiting for a resource that is locked globally.

RN(-NONE)　The program attempted to allocate a resource number and none was available.

SC　　　　The program is scheduled (waiting for CPU time) and will execute when all higher priority programs are suspended or dormant.

SP　　　　Spool/class buffer limit suspended.

SR　　　　The program is suspended waiting for another program to exit a shared subroutine.

SS　　　　The program has been suspended with SS command. You can reschedule the program with GO command, but parameters cannot be passed to the program.

TM　　　　The program has requested a timed delay of its own execution (EXEC 12 call, with name = 0).

WT(*progb*)	The program scheduled *progb* with wait (EXEC 9 or 23) and will continue execution when *progb* completes.
XS	The program is waiting for XSAM to become available. XSAM is required when using signals or UDSPs.
XQ	The program is executing.

PU (Purge Files)

Purpose: Purge files.

Syntax: PU *mask* [OK]

   mask   File descriptor.

   OK   Suppresses user prompt.

PWD (Display Working Directory)

Purpose: Displays present working directory.

Syntax: PWD
            PWD [-p]   (VC+ Only)

   -p   Only available when $VISUAL is set to a ksh-style command line editing mode. Displays the physical name of the current working directory.

       When $VISUAL is set to anything other than a ksh-style editing mode, PWD reports the physical name of the current working directory. The −P option is ignored in this case.

RESIZE (Set $LINES/$COLUMNS; VC+ Only)

Purpose: Set $LINES and $COLUMNS environment variables by querying an HP terminal for its physical screen size.

Syntax: RESIZE

RESIZE uses the following return variables:

$RETURN1 = 0   command executed successfully. $COLUMNS, $LINES, $RETURN2, and $RETURN3 are set to the values obtained from querying the terminal.

         = 1   error when querying terminal. $COLUMNS and $RETURN2 are set to 80. $LINES and $RETURN3 are set to 24.

         = 2   error when setting environment variables. $COLUMNS and $LINES are not set. $RETURN2 and $RETURN3 are set to the values obtained from querying the terminal.

         = 3   error when querying terminal and setting environment variables. $COLUMNS and $LINES are not set. $RETURN2 is set to 80 and $RETURN3 is set to 24.

$RETURN2   The width of the physical screen.

$RETURN3   The height of the physical screen.

RETURN (Return from Command File)

Purpose: Return to the previous level of command file nesting or to the interactive mode.

Syntax: RETURN [, *return1* [, *return2* [, *return3* [, *return4*
                                    [, *return5* [*return_s*]]]]]]

return1 Integer return status indicating success or failure of command file. A value of zero indicates success; a nonzero value indicates failure. If omitted, return1 is set to zero.

return2-5 Four integer values made available to return additional status information. Each omitted parameter is set to zero.

return_s A string of up to 256 characters for VC+, 80 characters for non-VC+; if omitted, return_s is set to null.

All the parameters for the RETURN command are position dependent; therefore, you must include commas to mark the positions of any omitted parameters.

The values returned are available in the predefined variables $RETURN1 through $RETURN5, and $RETURN_S, defined at the end of this section.

CI always executes a RETURN command when the end of a command file is reached, whether or not you included the command at the end of the file.

RM (Remove Files or Directories; VC+ Only)

Purpose: Removes files or directories.

Syntax: RM [-F|-I] [-QV] *file*|*mask* ...
              RM [-F|-I] [-QV] -R *directory*/ ...

-F   Purge files or directories without prompting for
              confirmation, regardless of the permissions of the file.

-I   Issue a prompt requesting confirmation before
              removing each file.

-R   Recursively purge the entire contents of directory and
              then purge the directory itself.

-Q   Quiet mode; inhibit error reporting.

.-V   Verbose mode.

--   End of the options; required if *mask* begins with "–".

file|*mask*   One or more files to be purged.

directory/   One or more directories to be purged. A directory may
              be specified with a trailing slash (*dirname*/) or with the
              .DIR type extension (*dirname*.dir). The trailing slash
              or the type extension is necessary when a file name
              without a type extension exists that has the same name
              as the directory.

RN (Rename File or Directory)

Purpose: Rename a file, directory or subdirectory

Syntax: RN *mask1* *mask2*

mask1   Current name.

mask2   New name.

RP (Restore Program File)

Purpose: Establish a program or prototype ID segment.

Syntax: RP *file* [*newname* [*options*]]

file   File descriptor of program (type 6) file to be restored. The first five characters of the file name are used as the program name, unless the optional parameter is specified.

newname   New program name to be used.

options   A character string that contains "C", "D", "P", "T", or "CP" (that is, both C and P) to select the following options:

C   Create a clone name if the specified or assigned name is already assigned to an RP'd program.

D   Create a prototype ID segment, *not* a program ID segment, for the program file.

P   Create a permanent program ID segment that will not be released when the program terminates. This is the default if *no* options are given.

T   Create a temporary program ID segment that will be released when the program terminates. This is automatically added to any other option that is given.

RS (Restart Program)

Purpose: Restart a program.

Syntax: RS [*prog*[*/session*]] [*pram*5]

prog   A 5-character program name that is to be OF'd and restarted. If not specified, the default is your session's first program.

/session   Session in which the program resides. Defaults to your session.

*pram*5   Parameters to be passed to the program. This can be five parameters or one long character string. The maximum runstring length, including the RS command and delimiters, is 256 characters.

Only a superuser can restart a program in another session or a system utility. The RS command is available only in CM.

RU (Run Program)*

Purpose:	Immediately schedule a program for execution and wait for its completion.
Syntax:	[RU] *prog*|*file* [*pram**5]
RU	Required only if the program name can be interpreted as a CI command or a command file name.
prog|*file*	A five-character program name or a file descriptor.
pram*5	Parameters to be passed to the program. The maximum runstring length, including the implied RU and delimiter, is 256 characters. This can be five parameters or one long character string.

SET (Display or Define Variables)

Purpose:	Display all positional, user-defined, and predefined variables, or define a user-defined or predefined variable.
Syntax:	SET [*var_name* = *string*] SET [-x⌉+x] [*var_name* [[=] *string*]] (VC+ only)
-x	Exports a defined variable or defines a new one as exported (also known as being in the environment). Exported variables are available to all copies of CI in the session, including CM.
+x	Imports an exported variable or defines a new variable as imported (also known as being a local variable). An imported variable is defined only for the current CI.
	If neither the −x or +x option is given, the variable is defined as a local variable.
var_name	A string of up to 16 or 32 letters, digits, and underscores, not starting with a digit. For non-VC+, the maximum string size is 16; for VC+, it is 32.
string_value	
	A string terminated by the end of the command line or a ";" whichever comes first. Command lines can be 255 characters in VC+, 83 characters in non-VC+.

---

* This command can also be entered in response to the System> or RTE: prompt.

SP (Spool Command; VC+ Only)

Purpose: Initiate and control spooling for I/O devices.

Syntax: SP [*command*]

If a command is not specified, the spool system enters interactive mode.

SP commands:

IN   Starts up the spooling system.

ON[,-S *sess*][,*lu*][,[,*options*]]

> Starts spooling for *lu* (default is *lu*=6). Subsequent output to this LU is written into the temporary spool file OUTSPOOL*xx*.SPL (*xx* is a spool system code). When spooling is stopped, the file is sent to the LU and the spool file is purged. Valid options are NC, NF, BP, and SS as defined at the end of the command descriptions.

ON[,-S *sess*][,*lu1*],*lu2*[,DC]

> Starts LU redirection from *lu1* to *lu2* (default is *lu1*=6). All I/O to *lu1* will be routed to the *lu2* specified. Both *lu1* and *lu2* must be valid LU numbers. By default, up to 2 levels of I/O redirection are allowed. For example, "sp,on,7,9" and "sp,on,9,11" will redirect LU 7 to LU 11.
>
> The DC (do not chain redirections) option can be specified to inhibit chaining of redirection.

ON[,-S *sess*][,*lu*],*file*[,*options*]

> Starts spooling for *lu* (default is *lu*=6). Subsequent output to *lu* is written in the file specified. At completion, the file is removed from the spool system, but not purged. Valid options are NC, NF, BP, SS, KC, and PU as defined at the end of the command descriptions.
>
> Unless KC is specified, spooled output written to the specified file does not contain EXEC CNTWD flags that were given in the spooled write requests. The file also does not contain device control requests that were issued to the LU via an EXEC 3.
>
> -s *sess* can be specified by a superuser for the ON command to start spooling for the given session number, *sess*. By default, spooling is started for your session.

LI , *file* [, *lu*] [, *options*]

> Queues file in the spool system for output to *lu* (default *lu*=6). Valid options are NC, NF, BP, SS, KC, and PU as defined at the end of the command descriptions.

RE , *fileref* [, *offset*]

> Restarts spool file output. *fileref* may be either the file name or the reference number displayed by the ST command. *offset* is a positive or negative number of relative records to skip.

PU , *fileref*   Removes the file from the spool system. This file is purged only if it is a spool file (that is, OUTSPOOL*xx*.SPL). *fileref* may be either the file name or reference number displayed by ST command.

OF [, *lu*] [, *session*]

> Stops spooling for *lu* (default *lu*=6). If *lu* was spooled to a default spool file, output is started to the device. Note that non-default spool files are not output. See the SP LI command to output non-default spool files.

> A superuser may terminate spooling or redirection for another session by specifying the *session* parameter.

LO , ON , *file*   Initiates error logging to file specified. (Superuser command.)

LO , OFF   Terminates error logging.

QU   Terminates the spool system. All spool system activity stops. Current spool files are purged and other files are removed from the spool system. (Superuser command.)

? or ??   Displays all spool commands.

ST   Status of spool system.

LR [, *sess* | *]

> Displays LU redirections. If *sess* is specified, redirections for that session number are displayed. If '*' is specified, redirections for all sessions are displayed. By default, redirections for your session are displayed.

LQ   Displays the spool system queue in quick format, one line per entry.

EX   Terminates the interactive mode.

The options that can be used with the ON and LI commands are:

NC	no carriage control
NF	no form feed
KC	keep control headers
PU	purge file when done
BP	print banner page
SS	suspend before and after printing

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 *prog* [*size* [*msegSize*]]

    prog   Program name.

    size   Program size in pages for non-VMA programs or the EMA size for EMA programs, not including PTE. Range is 2 ≤ size ≤ 1022/32733 for EMA size. Refer to "Models of EMA/VMA" in Chapter 10.

    msegSize   New MSEG size for EMA programs. Range is 1 ≤ *msegSize* ≤ 30.

---

* This command can also be entered in response to the System> or RTE: prompt.

TM (Display or Set System Clock)*

Purpose: Display or set the system clock.

Syntax (from CI>):

  TM [*month day year hr:min* [:*sec* [AM|PM]]]

   month Jan to Dec
   day 1 to 31
   year 1976 to 2037
   hr   0 (default) to 23
   min 0 (default) to 59
   sec 0 (default) to 59
   AM|PM AM (default) or PM

Display format (from CI>): Wed Oct 27, 1993 11:25:33 pm

Syntax (from System> and RTE:):

  TM [*hours min sec month day year*]

   hours 0 (default) to 23
   min 0 (default) to 59
   sec 0 (default) to 59
   month 1 to 12
   day 1 to 31
   year 1976 to 2037

Display format (from System>/RTE:): 11:37:13 SEP 21, 1992 MON

Time can only be specified in 24-hour format and all parameters must be specified when setting the system time. Setting the system time has the same effect as when setting from CI.

TO (Display/Change Device Timeout)

Purpose: Display or set timeout limit for a device.

Syntax: TO *lu* [*interval*]

 lu   LU number of device.

 interval Timeout value for device LU (in 10-ms intervals).
     $0 \leq interval \leq 65534$. If *interval*=0, device does not
     timeout.

---

* This command can also be entered in response to the System> or RTE: prompt.

TOUCH (Update File Times; VC+ Only)

Purpose: Updates the create, access, or update time in a file's directory.

Syntax: TOUCH [-amcb] [+b] [-r *ref_file*] [-t *time*]
                                 [--] *file* | *mask* ...

-a   Update only the access time of the file.

-b   Clear the backup bit of the file.

+b   Set the backup bit of the file.

-c   Do not create the file if the file does not exist.

-m   Update only the create time of the file.

-r *ref_file*
                 Use the corresponding create, access, and/or update time of *ref_file* instead of the current time.

-t *time*   Use the *time* argument as the create, access, and/or update time rather than the current time. The *time* argument is a decimal number in the following form:

$$[[CC]YY]MMDDhhmm[.SS]$$

where:
CC	is the first two digits of the year.
YY	is the second two digits of the year.
MM	is the month in the range 01 to 12.
DD	is the day in the range 01 to 31.
hh	is the hour in the range 00 to 23.
mm	is the minute in the range 00 to 59.
SS	is the second in the range 00 to 59.

If neither *CC* nor *YY* is given, the current year is used. If *YY* is specified, but *CC* is not, *CC* is determined as follows:

If *YY* is 69−99   then   *CC* = 19
If *YY* is 00−68   then   *CC* = 20

--   Signals the end of the options. This is only required if the first mask specified after the options begins with a hyphen (−).

file | *mask* ...
                 File descriptor of the file whose create, access, or update time is to be changed.

TR (Transfer to Command File)

Purpose: Transfer control to a command file.

Syntax: [TR] *file* [*pram*9*]

TR Transfer command. Required only if the file parameter can be confused with a CI command or a program (type 6) file descriptor.

file File containing CI commands.

*pram*9* One to nine parameters can be specified. They replace the positional variables $1 through $9 in the command file. (Default to zero-length strings.)

UL (Unlock Shareable EMA Partition; VC+ Only)*

Purpose: Unlock a shareable EMA partition.

Syntax: UL *label*

label Identifies a shareable EMA partition label, up to 16 characters.

UNALIAS (Delete Alias; VC+ Only)

Purpose: Deletes one or more aliases from CI and/or the Environment Variable Block (EVB).

Syntax: UNALIAS *alias1* [, *alias2* ...]

alias1,
alias2,... Strings of up to 32 letters, digits, and underscores, not starting with a digit.

If the user session has an EVB, CI first looks for the local alias and deletes it. If there is no local alias with the given name, CI looks for an exported one and deletes it.

UNPU (Unpurge Files)

Purpose: Recover purged files.

Syntax: UNPU *mask*

mask A file mask that specifies files to unpurge. A file can only be unpurged if its space has not been reused. "DL,@.@.P" lists purged files.

---

* This command can also be entered in response to the System> or RTE: prompt.

UNSET (Delete User-Defined Variable)

Purpose: Deletes one or more CI string variables or functions from CI and/or the EVB.

Syntax: UNSET *variable*

                   UNSET *variable1* [*variable2* ...] (VC+ only)

                   UNSET -f *function1* [*function2* ...] (VC+ only)

variableN, String of up to 32 letters, digits, and underscores, not
functionN   starting with a digit. The variable or function must exist.

-f   Specifies a CI function rather than a variable.

The UNSET command deletes a variable or function that you defined earlier in the session. You cannot use the UNSET command to delete positional and predefined variables except for $DATC, $IFDVR, $LINES, $COLUMNS, and $EVB_SIZE.

If the session has an EVB, CI first looks for the local variable or function and deletes it. If there is no local version, CI looks for an exported one and deletes it.

UP (Up a Device)*

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

Syntax: UP *lu*

lu   LU 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. $32 \leq vsSize \leq 65536$.

---

* This command can also be entered in response to the System> or RTE: prompt.

WC (Line, Word, Character Count)

Purpose: Count lines, words, and characters in the named file or files. Also keeps a total count for all named files. This command supports redirection.

Syntax: WC [*-options*] *file* | *mask* . . .

   options   One or more of the following:

   -l   Reports the number of lines in a file.

   -w   Reports the number of words in a file.

   -c   Reports the number of characters in a file.

   -q   (Quiet) Inhibits error reporting.

   --   Marks the end of the options. This is only required when the mask begins with a "−".

   The l, w, and c options can be used in any combination to report a subset of lines, words, and characters. The default is −lwc.

   file | *mask*   One or more files for which you want the line, word, or character count.

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 current command stack file.

WH (System Status Reporting)

Purpose: Display system information.

Syntax: WH [-P] [*option*] [*destlu*]

-P   Causes WH to generate "More..." prompts at the end
          of each screenful on long listings.

option   Specifies the information to be displayed:

none	all programs for current session or terminal
AC	all active (non−dormant) programs
AL	all restored programs
CL	class numbers
D	XSAM proto-ID segments
OP *prog*	program specified by <prog>
PA	memory partitions
PL *progmask*	
	programs with names selected by *progmask*
RN	resource numbers
SC	scheduled and I/O suspended programs
SE	all active sessions
SESS#	programs in that session
SH	shareable EMA
ST	system status
US	all user programs

destlu   specifies LU (if other than scheduling terminal) to send
          output.

Note that WH executes while the state of the system is changing and,
therefore, some reported information might be incomplete,
inaccurate, or duplicated.

WHILE-DO-DONE (Control Structure)

Purpose: Allow repeated execution of a group of commands.
               Can be used only in a command file.

Syntax: WHILE *cmd-list1*
              DO *cmd-list2*
              DONE

cmd-listn   A list of commands either one command per line or
              multiple commands per line separated by semicolons.
              A command-list can be null.

WHOSD (Report Users of File/Directory or Volume; VC+ Only)

Purpose:	Report users of a CI file/directory or a volume. This command supports redirection.
Syntax:	WHOSD [-t] [-m *mask*] *file*|*dir*|*lu*
-t	Trace ID segments and proto-IDs back to a file name.
-m *mask*	Check only those ID segments and proto ID segments whose names match the mask.
file|*dir*|*lu*	Specifies the file, directory, or volume of which you want to report the user.

WS (Display/Change VMA Working Set Size)*

Purpose:	Display or modify VMA working set size of a restored program.
Syntax:	WS *prog* [*wsSize*]
prog	Program name.
wsSize	Working set size in pages (not including PTE). $2 \le wsSize \le 1022/32733$. Refer to "Models of VMA/EMA" in the VMA/EMA chapter.

XQ (Run Program Without Wait)*

Purpose:	Schedule a program for execution, then return to CI.
Syntax:	XQ *prog*|*file* [*pram*5*]
prog|*file*	Program name or file descriptor.
pram*5	Parameters to be passed to the program. The total runstring has a limit of 256 characters.

? (Help)

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

---

* This command can also be entered in response to the System> or RTE: prompt.

$1 – $9 (Positional Variables)

Nine positional variables 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.

User-Defined Variables

A user-defined variable name is a string of up to 16 letters, digits and underscores not starting with a digit. The SET command defines both the variable name and its contents. When you reference the variable after it has been defined, precede the name with a $ sign and follow it with a space or any other character that is not a letter, a digit, or an underscore. A user-defined variable can be deleted using the UNSET command.

Environment Variables (VC+ Only)

Environment variables allow programs within a session to share variables. This is an extension of the SET command in CI. The original SET command can be thought of as resulting in "local" (to one copy of CI) variables. The extension creates "session" variables that can be "exported" to (and "imported" from) the session environment. Exported variables are available to all programs in the session. It is possible to have a local and an environment variable by the same name. Access is always to the local variable.

The SET command by itself displays all defined variables, in alphabetical order.

The UNSET command is used to remove the variable. When the UNSET command is given, first the local variable list is searched; if a local copy is found, it is unset. If no local variable is found, the exported variables are searched. Unsetting an exported variable removes it for all programs in the session.

Predefined and Other Variables

When you begin a CI session, there are predefined variables. These variables are initialized to default values by CI. However, you can use the SET command to modify the values of all the variables except $MY_NAME. Also, $WD is different in that you can modify it, but CI updates it after each CD or WD command. With VC+, some variables are automatically exported and cannot be imported. The SET and ECHO commands can be used to display the values of predefined variables. Unless otherwise noted, you cannot delete these variables.

$AUTO_LOGOFF

> Allows for automatic logoff if session is inactive. CI initializes $AUTO_LOGOFF to zero, which means automatic logoff is not in effect. If you set $AUTO_LOGOFF to a nonzero value, CI times out after that many terminal timeouts. If CI is the only active program, after four CI timeouts, an EX,B command is executed to terminate the session.
> Default: 0
> Allowed: ranges from 0 to 32767

$CMNDO (VC+ only)

> This variable can be set to TRUE or FALSE to control CMNDO usage in programs that support the CMNDO monitor. Refer to the *RTE-A User's Manual*, part number 92077-90002, for information on CMNDO.

$COLUMNS (VC+ only)

> The number of columns in your display. This variable is used by LI, LS, and FmpPaginator. CI automatically exports it at startup. You may import, modify, or unset $COLUMNS. If the value is invalid, CI uses the default. The RESIZE program can be executed to compute the size of your display and set $COLUMNS accordingly. $COLUMNS can be deleted by the UNSET command.
> Default: 80

$DATC   The datecode revision of the operating system; for example, 6100 for Revision 6.1. This variable is for user information and can be deleted by the UNSET command.

$EVB_SIZE (VC+ only)

> The size of the session's EVB in pages; zero if no EVB is available. This variable is for user information and can be deleted by the UNSET command.

$FRAME_SIZE

Size of command stack display in lines. When you log on, the command stack display size is initialized to 20 lines.
Default: 20
Allowed: ranges from 1 to 32767

$HOME
For VC+ CI, $HOME is your home directory, as given by User Defined Search Path (UDSP) #0, set up by the PATH program or CI. If PATH is not run, $HOME is set to the directory in which the first copy of CI starts up. $HOME is automatically exported and cannot be imported or deleted by UNSET.

For non-VC+ CI, $HOME is set to the directory in which CI starts up.

$IFDVR (VC+ only)

The interface driver of your port; for example, IDZ00, ID100, ID400, or ID800. This variable is for user information and can be deleted by the UNSET command.

$KILLCHAR (VC+ only)

Used with the visual editing modes. When the kill character is entered, the contents of the current command line become invalid and the user is given another prompt. When not defined by the SET command, the DEL character is used.

$LINES (VC+ only)

The number of lines in your display; used by LI, LS, and FmpPaginator. CI automatically exports $LINES at startup. You may import, modify, or unset it. If the value is invalid, CI uses the default. The RESIZE program can be called to set the value correctly for your display. $LINES can be deleted by the UNSET command.
Default: 24
Allowed: ranges from 0 to 32767

$LOG
A flag indicating if commands executed in a command file are logged to the terminal. CI initializes this variable to OFF, which means that commands are not displayed at the terminal. To display commands at the terminal, set the value to ON.

$LOGON
The user and group name with which the user is associated during the session in the form *"user.group"*.

$MY_NAME The true or schedule name of CI. This variable can never be altered.

$OLDPWD Set to the previous working directory ($WD) whenever a WD or CD command is executed.

In VC+ CI, $OLDPWD is automatically exported and cannot be imported or deleted by UNSET.

$OPSY The value of the system entry point $OPSY for the current system. For example, the value of $OPSY for RTE-A Revision 6000 is -125.

$POLL Set by the POLL command to be the command that is executed via the POLL command. $POLL can be altered by the SET command and can be deleted by the UNSET command. $POLL is not predefined.

$POLLINT The approximate number of minutes between the successive executions of $POLL. $POLLINT cannot be altered by the SET command but can be deleted by the UNSET command. $POLLINT is not predefined.

$PROMPT The prompt that is displayed when CI is waiting for input. CI initializes this variable based on the program name with which CI has been scheduled, for example, "CI..A>".
Default: CI>
Allowed: up to 78 chars in VC+; up to 16 in non-VC+

$REPROMPT (VC+ only)

If this variable is not set or is TRUE, CI automatically reissues the prompt upon a timeout. This can be inconvenient to workstation users using terminal emulators that de-iconify themselves when they receive output (that is, hpterm with mapOnOutput set to True). To correct this behavior, you can either set the terminal's timeout to 0, which is not recommended, or set REPROMPT to FALSE. If REPROMPT is set to FALSE and the terminal emulator still de-iconifies itself when a timeout occurs, change the interface driver protocol to not use DC1 handshakes (for example, use Xon/Xoff protocol). This variable is not predefined.
Allowed: TRUE or FALSE

$RETURN1 – $RETURN5

Five integer values (ASCII representation) returned from execution of the last command. CI updates the values as commands are executed. These variables can be set to values between -32768 and 32767, inclusive.

$RETURN_S A character string returned from execution of the last
command. The string is 256 characters in VC+, and 80
characters in non-VC+. CI updates the value as
commands are executed.

$RU_FIRST A flag indicating whether RU or TR is to be assumed if
you enter only a file name in response to a CI prompt
or as a line in a command file. $RU_FIRST can have
the following meanings:

TRUE   CI first attempts to execute an RU
            command for the specified file name.

FALSE   CI first attempts to execute a TR
            command for the specified file name.

You should set the variable to FALSE if you execute
more command files than program files.
Default: TRUE

$SAVE_STACK

A flag indicating if the command stack is saved when
you exit CI or when the command stack file is changed
with the WD command. $SAVE_STACK can have the
following values:

TRUE   the stack should be saved.

FALSE   the stack should not be saved.

Default: TRUE

$SESSION Number of your current session. CI initializes this
variable to your session number and updates the value
after execution of every CI command.

$VISUAL (VC+ Only)

> Required to edit the command line in a manner similar to ksh or csh, command interpreters that are used in the HP-UX operating system. Values can be CSH, EMACS, GMACS, VI, or CI, with or without the NODUPES option. This variable is not predefined.

> Setting $VISUAL to EMACS, GMACS, VI, or CSH effects the functionality of the RTE command stack. By default, the RTE command stack routines do not insert duplicate lines in the stack. When setting $VISUAL, duplicate lines are allowed in the stack. To override this behavior, use the NODUPES option. For example, to use the VI editing mode without saving duplicate lines in the command stack, set $VISUAL to "vi,nodupes".

> Executing the command "set visual ci,nodupes" is equivalent to not setting the variable. Unsetting it accomplishes the same thing.

> Allowed:

emacs,[nodupes]	ksh-style EMACS command editing
gmacs,[nodupes]	ksh-style GNU EMACS command editing
vi,[nodupes]	ksh-style VI command editing
csh,[nodupes]	csh-style name completion and directory lists
ci,[nodupes]	disables command editing, except for "/" commands

$WD   Name of the current working directory. CI updates this variable after execution of each WD or CD command.

4

EDIT/1000 Commands

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

filedes   File descriptor of the file to be edited.

commands Edit commands to be executed upon entering EDIT.
            Multiple commands are separated by a command
            separator "|" (default).

EDIT prompt character "/" (default).

Separate elements in the runstring or the command by a space or a
comma.

Line Specification

.   Current pending line.

$ or >   Last line in the file.

n   Line number.

[*line spec1*][*line spec2*]
            Line range specifications (absolute line numbers or an
            offset from the current line).

*   First line of *line spec*.

^*n* or –*n* Backward *n* lines.

+*n*   Forward *n* lines.

:*x*   Marked line, where marks (indicated by *x*) are A
            through Z.

'*pattern*' Search forward to line containing specified pattern.

'*pattern*' Search backward to line containing specified pattern.

Use a space or comma between commands or parts of commands that
might otherwise be ambiguous.

Edit Session Options (SE Command)

SE,*option*[,*value*]   Sets various EDIT session options and defaults.

EDIT session options:

AC	anchor character (default=^)
AS	verify dangerous commands (default=on)
BE	bell with prompt (default=off)
CD	set line editing graphics submode within screen mode (default=off)
CF	case folding (default=on)
CS	command separator (default= |)
DF	display functions in screen mode (default=on)
EC	escape character (default=\\)
FL	fill columns: first, last, end (1, 70, 256)
IC	indefinite character (default=@)
IN	save indentation on fill (default=off)
LE	maximum characters on line (default=256)
PC	EDIT prompt character (default=/)
QU	quiet option (default=off)
RE	regular expressions (default=off)
RT	return to pending line after multiple search (default=on)
SD	screen size (default=10, 10 ,2)
SL	screen length (default=30 or 100)
SW	screen width (sensed when needed; SW*n* sets screen width to *n*)
TC	tab character (default=TAB key or ctrl-I)
TS	time stamp update (default=on)
VW	vertical window (default=10, 10)
WC	window columns (default=1, 256)

Screen Edit Commands

*[line spec1][line spec2]*S   Start screen edit.

Screen Mode Control Commands

Control commands entered twice leave the workfile unchanged.

ctrl-Q	Save screen edits and quit screen mode.
ctrl-U	Same as ctrl-Q; use for terminals with Xon/Xoff handshake protocol.
ctrl-P	Go to previous screen.
ctrl-R	Same as ctrl-P; use for X.25 pad terminals.
ctrl-F	Forward one screen.
ctrl-S	Start next screen.
ctrl-T	Same as ctrl-S; use for terminals with Xon/Xoff handshake protocol.
ctrl-X	Start next screen with larger (maximum) screen size.
ctrl-C	Execute command in line mode and return to screen mode.
ctrl-D	Enter ctrl-D mode (see below).
esc-4-ctrl-B	Break line at cursor position.
ctrl-J	Join following line to this one.
ctrl-O	Duplicate line on the screen.
ctrl-A	Move to left end of current line.
ctrl-Z	Move to right end of current line.
ctrl-K	Mark line; enter alphabetic character at colon.

Ctrl-D Mode (Line Graphic Editing)

A *[option]*	Arrow. Draw a path through defined points and put an arrow head at end of path.
B *[option]*	Box. Draw a closed polygon through defined points.
C	Copy.
E	Erase path (same as P E command).
L *[option]*	Lines. Take point list as pairs of points and draw lines between the pairs.

M   Move. Same as copy, but erase original area.

P [*option*] Path. Start at the first point in the list and draw a line
           through each consecutive point until end of the list is
           reached.

Q   Quit ctrl-D mode.

R   Re-mark. Restore marks of previous ctrl-D mode
           command and re-enter ctrl-D mode. If the previous
           command was a copy or move, the locator points are
           not restored.

U   Undo. Undo the last ctrl-D mode command.

where *option* can be one of the following:

N   No read. Can save time if no screen mode changes
           have been made.

B   Bold line style.

E   Erase.

P   Paired line style.

Display Commands

?[*command*]

>or

H [*command*]

>or

HE [*command*]

>Display information about the specified command (default=summary of commands).

?[*option*]

>or

H [*option*]

>Display the requested information.

>Options:

>| | |
>|---|---|
>| AB | listing of EDIT abort messages. |
>| DA | revision code of the EDIT program being run. |
>| EX | explanation of EDIT abbreviations. |
>| LS | line specification explanation. |
>| PA | pattern explanation. |
>| PL | pending line edit information. |
>| RE | regular expression explanation. |
>| RM | recover mode explanation. |
>| RO | description of EDIT runstring options. |

>When "[more . . ." is displayed at bottom of a help screen, pressing RETURN displays the rest of the current help information without pause. Pressing any other key causes the next help screen to display. Type "a" to abort help listing.

HL[P]   Display ruler header line. P makes ruler same length as pending line.

??   Display current version of EDIT program and current file.

[line spec1][line spec2]L[max] [+][list file][/]

    Display a maximum number of lines (default = 20).

    The lines can be listed or appended (+) to a list file. Enter '/' to suppress OK? prompt.

L   Display 20 lines plus next pending lines.

[line spec1][line spec2]LN[max] [+][list file][/]

    Display 20 lines with line numbers plus next pending line.

[line spec1][line spec2]LU[max] [+][list file][/]

    Display lines without line numbers (turn off LN).

[line spec1]LE

    Display the number of characters in the current pending line or specified 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 (if *option* is not specified, summary of all EDIT options is displayed). *option* may also be:

        FM   show whether changes have been made.
        MA   display current text marks.
        SW   display current screen width.
        UN   display the Undo List maintained by EDIT.

SH UN [n1, n2]

    Display the commands that would be used by the UN command to undo the last change.

[line spec1]SZ

    Display approximate number of words in the destination file.

[line spec1][line spec2]W

    Display window of lines and return to pending line.

WN | WU   Display window of 20 lines with line numbers (WN) or without line numbers (WU) and return to pending line.

/   Display command stack.

Line Edits

*[line spec1]*I*<line char edits>*

> Insert text before specified line (default is pending line).

{*<space>*} *<text>*

> Add text after pending line.

*[line spec1][line spec2]*K*[max][+][list filedes][/]*

> Delete the specified lines (default is pending line). The deleted lines can be saved (or appended if +) to a list file. Enter '/' to suppress OK? prompt.

*[line spec1]*C*<line char edits>*

> Edit pending line then advance pending line.

*[line spec1]*O*<line char edits>*

> Duplicate the specified line and edit copied line. Display edited line (default is pending line).

*[line spec1]*P*<line char edits>*

> Edit the specified line and display (default is pending line).

*[line spec1]*Q

> Edit pending line with terminal edit keys.

*[line spec1]*R*<text>*

> Replace the specified line with text (default is pending line).

*[line spec1]*J

> Join the specified line to the following line (default is pending line).

Line Mode Character Edits

ctrl-B	Break line at cursor position.
ctrl-C	Delete characters.
ctrl-R	Replace characters.
ctrl-S	Insert characters.
ctrl-P	Insert characters (frees ctrl-S to be used with Xoff).
ctrl-T	Truncate line at cursor position.
ctrl-X	Extend line, adding characters to end of line.
\	Escape character.
ctrl-\	Non-printing escape character.
/	Enter current prompt (default = /) with any control character edit to preserve character in that position.
<tab>	Skip to next tab stop, leaving skipped characters unmodified.
<text>	Replacement text to be entered on pending line.

Search Commands

*[line spec1][line spec2]***B***/pattern/* [A,N,Q,V]

> Find a line with *pattern* (default = last pattern specified) within the specified range (default = line 1 to EOF).

*[line spec1][line spec2]***F***/pattern/*[A,N,Q,V]

> Find a line with *pattern* (default = last pattern specified) within the specified line range (default = pending line to EOF).

*[line spec1][line spec2]***D***/pattern/* [A,N,Q,V]

> Delete lines from the specified line (default = current line) to the line containing the specified pattern (default = last pattern specified).

' *pattern*   Search forward for a line containing *pattern*.

' *pattern*   Search backward for a line containing *pattern*.

Search Options

A   Multiple (All) search
N   No-window parameter
Q   Suppress display (Quiet)
V   Reverse match

Exchange Commands

*[line spec1][line spec2]*G/*match pattern/substitute*/[N,R,S]

> Exchange *match pattern* with *substitute* over the
> specified range without listing (default = pending line).

*[line spec1]*Y/*match pattern/substitute*/[N,Q,R,S]

> Exchange *match pattern* with *substitute* on the specified
> line and display the next occurrence
> (default = pending line).

*[line spec1][line spec2]*U/*match field/substitute*/[Q]

> Unconditional exchange over specified range of
> number of characters in *match field* with *substitute*.

*[line spec1][line spec2]*X/*match pattern/substitute*/[N,Q,R,S][/]

> Exchange *match pattern* with *substitute* over the
> specified range (default = pending line). '/' suppresses
> OK? prompt.

Exchange Options

N   No-window parameter
Q   Suppress display (Quiet)
R   Remove zero-length records
S   Single exchange parameter (one exchange per line)

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 work area with another file. '/' suppresses OK? prompt.

$$[\#lines]$$
*[line spec1]***M** *filedes* *[start line]*   or
$$[:stop\ line]$$

Merge in file specified after pending line.

WC *filedes*   Create file without exiting EDIT session.

WR *[filedes]*   Replace (write over) file without exiting EDIT session. Default is current file.

Terminations

A[/]   Abort EDIT leaving source file unchanged. '/' suppresses OK? prompt.

AS[/]   Abort EDIT and save work file. '/' suppresses OK? prompt.

EC *filedes*   Create a file and end EDIT session.

ER   Replace (write over) source file and end EDIT session.

ER *filedes*   Replace (write over) specified file and end EDIT session.

Control Commands

<space>text   Append a line of text following current pending
                 line.

#xxx[n1,[n2]]   Add sequence numbers with 3-character identifier
                 field *xxx* in columns 73 through 80. *n1* is the starting
                 number; *n2* is the increment number.

*<text>   Enter comment line in command file.

| [line spec1][line spec2]_[n][Q]
                 Repeat command *n* times. '|' is command
                 separator and must precede the repeat command.

*[line spec1][line spec2]**BC** startcol[:stopcol] [destcol] [Q]*
                 Block copy.

*[line spec1][line spec2]**BK***
                 Kill trailing blanks in specified line range.

*[line spec1][line spec2]**BM** startcol[:stopcol] [destcol] [Q]*
                 Block move.

*[line spec1][line spec2]**CO** [Q]*
                 Copy text.

*[.] [*] **FL** [Q] [R] [I]*
                 Fill text. 'R' removes indentation; 'I' leaves
                 indentation.

*[line spec1]**K**x*
                 Mark a line with character *x*..

*[line spec1][line spec2]**MO** [Q]*
                 Move text.

RU*,program*   Run program and return to EDIT at pending line
                 upon termination.

SC   Copy screen memory; insert after pending line.

T[*n1,...,n30*]   Set tab columns up to 30 settings.
TA   Set tab columns to 7 and 21 (for Assembly).
TF   Set tab columns to 7, then every 4 (for FORTRAN).
TL   Set terminal tab stops to line mode tabs.
TM   Set tab columns to 10,26,40,44,48 (for Macro).
TP   Set tab columns to every 3 (for Pascal).
TS   Set terminal tab stops to screen mode tabs.
TU   Set tab columns to 8 (UNIX default).

[*line spec1*]**TI***n*
>Add time and date to line starting at column *n*.

[*line spec1*][*line spec2*]**TK**
>Expand tab characters into spaces.

TR,*namr*,[Q][/]
>Transfer to a command file or an input device.
>Specify Q to suppress the listing; specifiy / to
>suppress OK? prompt.

UN
>Undo the command executed immediately before
>this command and revert file to the prior state.

[*line spec1*]**UY**[*n1*][*n2*]
>Return text removed via the Undo (UN) command
>to the text file.

/[*n*] *or*
//[/...]
>Command stack.

5

LINK Commands

LINDX Runstring

Purpose: Index library files for faster searching.

Syntax: LINDX, *inputFile*, *outputFile* [, +NL] [, +L *listfile*]

  inputFile File to be indexed.

  outputFile Indexed library (different from *inputFile*).

  +NL Specifies no list of entry points.

  +L *listfile* Sends the entry point listing to the specified file.

LINK Runstring

Syntax: LINK [*pram**n]

  pram*n Any number of parameters up to the 80-character
runstring limit. Options and/or file names can be
specified in any order. An option is prefixed with a
plus (+). A file name must include one of the
following type extensions:

    .REL Relocatable code.
    .R*nnn* Relocatable file where *nnn* is 3 integers.
    .LIB Relocatable library.
    .MAP Load map.
    .RUN Executable program.
    .SNP Snap file.
    .LOD LINK command file.
    .DBG DEBUG file.

If no parameters are specified in the runstring, LINK runs
interactively, ending when the EN or AB command is entered.

LINK Runstring Options

The LINK options that can be used in the runstring are (the following section contains detailed descriptions of the LINK commands that correspond to these options):

+B   Batch mode; LINK never goes interactive.

+CR:*file* | *crn*   Specify scratch disk file or cartridge (for VMA backing store file).

+DE   Create Symbolic Debug file for relocatable files included in the runstring.

+DM   Set Debug Monitor mode.

+DP   Prohibit purging of existing program file.

+E *cmd* | *cmd*...   Execute LINK commands from the runstring. If given, this must be the last option in the runstring; the remainder of the runstring is a series of LINK commands, separated by vertical bars (|). These commands are executed before interactive commands or transfer file commands are executed.

+EC   Echo commands, all input commands are sent to list file or device.

+LC   Use labeled system common.

+LL:*file* | *lu*   Specify list file or LU to which messages and the load map are sent.

+MA   Display load map on terminal.

+RO   Reorder modules in the data segment to reduce base page links.

+SC   Use blank system common.

+SP   Declare program as shareable (CDS programs only).

+SU[:OF]   Specify a system utility. The optional OF parameter indicates that any user can remove the program from physical memory.

+SZ : [+]*pages*   Set non-CDS program to specified number of pages
in physical memory. Pages can be set to a value be-
tween 1 and 32, inclusive.

+WD [: /*dir* [/ @>*dsinfo*]]
Set the LINK working directory for file references.

The colon is required when the CR, LL, SU:OF, or SZ option is used
in the runstring. The LINK runstring options can be entered in any
order, with the exception of the E option. Because the LC and SC
options are mutually exclusive, only one should be entered in the
runstring. If both are entered, the last one specified is used.

$VISUAL Command Editing within LINK

To use the $VISUAL command editing modes within LINK, set the
$CMNDO_LINK or $CMNDO environment variable as follows:

```
CI> set -x CMNDO_LINK = T    Use CMNDO from LINK,
                             but not other utilities.
or
CI> set -x CMNDO = T         Use CMNDO from all
                             utilities that support it.
```

If you have $CMNDO set to TRUE but do not wish to use CMNDO
from LINK enter:

```
CI> set -x CMNDO_LINK = F
```

LINK Commands

AB (Abort)

Purpose: Abort LINK immediately.

Syntax: AB

AL (All Memory Locked – CDS)

Purpose: Declare that at run-time, the operating system will bring all segments into memory before dispatching the program.

Syntax: AL [UN]

UN   Unlocks memory.

AS (Assign Partition)

Purpose: Assign a partition where the program will reside.

Syntax: AS *partNum* [C|D]

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

C|D   For CDS program, C specifies the code partition and D the data partition. This parameter is required for a CDS program and ignored for a non-CDS program.

BP (Report Base Page Usage)

Purpose: Report the number of base page links used by non-CDS modules.

Syntax: BP
          +BP (in runstring)

CD (Code Segment – CDS)

Purpose: Establish default code partition size by setting number of segment blocks per code partition. (A segment block is the length of the largest code segment. The load map contains this information.)

Syntax: CD [*segments*]

segments   Number of segment blocks per code partition, in range 1 to 128. Default is all segment blocks.

CR (Specify VMA Backing Store File)

Purpose: Specify backing store file or FMGR cartridge (crn) to be used by LINK.

Syntax: +CR:*file* | *crn*   (in runstring only)

file   Specifies the name of the LINK backing store file.

crn   Specifies the FMGR cartridge where LINK puts the backing store file.

DB (DBUGR)

Purpose: Append the DBUGR subroutine to the program file. (This command is provided for backward compatibility. Refer to the *RTE-6/VM Debug Subroutine Reference Manual*, part number 92084-90014, for information on the DBUGR program.)

Syntax: DB

DE (DEBUG)

Purpose: Create .DBG file for Symbolic Debug/1000.

Syntax: DE
             +DE (in runstring)

DI (Display)

Purpose: Display undefined externals.

Syntax: DI

DM (Debug Monitor)

Purpose: Set Debug Monitor mode.

Syntax: DM
             +DM   (in runstring)
             DM [OF]   (re-link syntax)

OF   Turns off debug monitor mode. Only valid during relinking.

DP (Do Not Purge)

Purpose: Inhibit purging of existing program files when running
             LINK interactively. EN or NA will override DP.

Syntax: +DP   (in runstring only)

EC (Echo)

Purpose: Echo loader command file commands.

Syntax: EC
             +EC   (in runstring)

EM (Extended Memory Access)

Purpose: Set EMA size.

Syntax: EM [*pages*]   [*model*]

pages   Number of pages of EMA space, in range 2 to 1022;
             maximum number of pages for Extended Model
             programs is 32733.

model   L   use Large EMA model.
             X   use Extended EMA model.

             If a model is not specified, the Normal EMA model is
             used.

EN (End Link)

Purpose: End command input.

Syntax: EN [*file*]

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

ES (EMA Segment)

Purpose: This command relocates a common block to the start of
                  an EMA segment, for use in programs that call the
                  RteAllocShema routine to attach shared EMA areas.

Syntax: ES , *commonblock* , *emaseg*

commonblock
                  The name of the common block that should be
                  relocated to the given EMA segment.

emaseg   An EMA segment number from 1 to 64.

FO (Force Load)

Purpose: Force linking of a program or overlay, regardless of
                  undefined externals.

Syntax: FO

HE (Heap Area – CDS)

Purpose: Set heap area size.

Syntax: HE [*words*]

words   Size (in words) of data partition heap area, in range 4
                  (default) to the remainder of the data partition.

IF (Conditional Execution of LINK)

Purpose: Selective execution for specified system.

Syntax: IF A | 6 *linkCommand*

A   Execute when linking on or for RTE-A.

6   Execute when linking on or for RTE-6/VM.

linkCommand
                  A valid RTE-A or RTE-6/VM LINK command.

LC (Labeled System Common)

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

Syntax: LC
+LC (in runstring)

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* | *lu*   (in runstring)

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.

LO (List Program Attributes)

Purpose: List program attributes during relinking.

Syntax: LO

MA (Display Load Map)

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

Syntax: +MA   (in runstring only)

ML (Memory Locked – CDS)

Purpose: Memory-lock current code segment when segmenting manually.

Syntax: ML
             ML *segmentNum* [UN]   (relinking syntax)

segmentNum

      Number of segment to be memory-locked (or unlocked). Default is current segment.

UN   Unlocks specified segment. Default is lock.

MS (Multiple Search)

Purpose: Search a library file.

Syntax: MS *file*

file   File descriptor of file to be searched.

NA (Name)

Purpose: Set program file name.

Syntax: NA [*file*]

file   File descriptor to be used for program file.

NS (New Segment – CDS)

Purpose: Start new code segment for manual segmentation.

Syntax: NS [*pages*]

pages   Maximum code segment length, in range 1 to 31 pages. Default is 30 pages.

OR (Order EMA Area)

Purpose: Specify order in which EMA areas are allocated.

Syntax: OR *area1 area2 area3* ...

  areaN   Name of EMA area declared in source file.

OS (Operator Suspend)

Purpose: Suspend LINK (allows access to CM> prompt) until you enter the system GO command.

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 for program output file.

PA (Page Align EMA Area)

Purpose: Start the next EMA area (on OR command) at page boundary.

Syntax: PA

PC (Program Capability)

Purpose: Set capability levels of program to be linked.

Syntax: PC *progcplv* [*rquscplv*]

  progcplv   Capability level assigned to the program.

  rqunscplv   Capability level required by a user to run the program. Default is 0.

PR (Priority)

Purpose: Set the priority of the program.

Syntax: PR *nn*

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

PS (Page Align Overlays -- non-CDS)

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.

RM (Relocate Module)

Purpose: Include a module as part of current segment.

Syntax: RM *file symbol*

  file   File descriptor of file to be included in program.

 symbol   Entry point for the module to be relocated.

RO (Reorder)

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

Syntax: RO
               +RO   (in runstring)

SC (System Common)

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

Syntax: SC
           +SC   (in runstring)

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 the system library files.

SH (Shareable EMA)

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

Syntax: SH *label* [*partNum*]

label   Partition name (up to 16 characters).

partNum   Reserved partition number, in range 0 (default, not reserved) to 1023.

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.

SP (Shareable Program – CDS)

Purpose: Declare this program to be shareable.

Syntax: SP [UN]
           +SP[:UN]   (in runstring)

UN   Declares program unshareable during relinking.

ST (Stack – CDS)

Purpose: Set size of stack area within data partition.

Syntax: ST [+ | –] *words*

words Size of stack area, in range 6 to a maximum number that is dependent on the heap size and MSEG usage.

If the number of words specified is preceded by a plus (+) or a minus (–), then *words* is added to or subtracted from the default number of words estimated by LINK.

SU (System Utility)

Purpose: Declare system program. (A system program cannot be cloned. Only a superuser can remove it, unless OF is specified.)

Syntax: SU [OF]
+SU[:OF]   (in runstring)

OF Allows any user to remove the program. Default is only superuser can remove program.

SZ (Size)

Purpose: Specify number of physical memory pages required to run program. For CDS programs, specifies the number of data partition pages.

Syntax: SZ *pages* | +*pages*
+SZ:*pages* | +*pages*   (in runstring)

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

+*pages* Number of pages, in range 2 to 32, to be added to pages required by program. Default is zero.

TR (Transfer)

Purpose: Transfer control to a file containing LINK commands.

Syntax: TR *file*

file File descriptor of command file. If no type extension specified, LINK assumes .LOD.

VM (Virtual Memory)

Purpose: Specify that program uses VMA; specify size of VMA backing store file. For backward compatibility, this command may also be entered as VS.

Syntax: VM [*pages*] [*model*]

pages Maximum number of pages in backing store file; in the range 32 to 65536, inclusive. Default is 8192.

model L use Large EMA model.
X use Extended EMA model.

If *model* is not specified, Normal VMA model is used.

WD (Default Working Directory)

Purpose: Provide a default directory and node name to be used when Link opens files.

Syntax: WD *directory* [/ @>*dsinfo*]
+WD [:*directory* [/ @>*dsinfo*]]   (in runstring)

directory Specifies the default directory path to use when opening a file. If this parameter starts with a / (slash), then the path starts at the global directory. Otherwise the path is relative to the current session's working directory.

@ A placeholder for the file name.

dsinfo DS transparency specifier of a remote node and optional user logon of the following form (note that if the user logon and password is specified, the brackets ([ ]) are required as part of the syntax):

node [*user* / *password*]

WS (Working Set Size of VMA)

Purpose: Specify working set size of VMA, excluding the PTE.

Syntax: WS *size*

  size   Number of pages, in range 2 to 1022 (default is 32); maximum WS size is 32733 pages for Extended Model programs.

* (Comment)

Purpose: Ignore remainder of line.

Syntax: *   (in column one)

? (Help)

Purpose: Display help information.

Syntax: ? [*linkCommand*]

  linkCommand

        Help on a particular command. Default is a summary of all commands.

6

Utilities

ASAVE and ARSTR

Purpose: Save or restore one disk LU, a group of disk LUs, or an entire disk unit to a tape.

Syntax: ASAVE [*command* [;*command*;...]]

ARSTR [*command* [;*command*;...]]

command One of the following (or its first two letters); default = interactive mode.

EXIT | END | ABORT

Terminate ASAVE

HELP | ? | ?? Online help.

LH [*fileNum*]

List header record for file specified by fileNum or closest file to it on current tape. Default = next save file. If save file for disk LU crosses tapes, it has the same fileNum.

LL [file | *lu*]

Open list file or LU for log information (in addition to terminal); close previous one. Default = close previous one.

RE *fileNumlu* [*fileNumlu* ...] [VE]

(ARSTR only) Restore a file from tape to disk. Default *fileNum* = 1,2,... Default *lu* = LU used during ASAVE. Option:

VE Verify

RW Rewind and set the tape offline. If a CTD, it will unload also.

SAVE *lu1* [*lu2* ...] [UN] [AP] [VE] [NL] [NC]

Save a disk to tape. Options:

UN Save entire disk unit containing LU 1.
AP Append files to current ASAVE tape.
VE Verify.
NL Do not lock disk LUs during save operation.
NC Do not checksum the data records on the tape.

TAPE [*lu*] Specify tape LU. Default = display tape LU.

TITLE [*text*]

(ASAVE only) Specify title ($\leq$ 40 characters) for
SAVE operation. Default = no title.

UE [ON|OFF]

Specify user error-handling mode (initially on).
Default = display current mode.

CALLM

Purpose: Build a single compressed input text file for the
                 CALLS utility program.

Syntax: CALLM [*-options*] *commandfile* *destfile*

-options   A string of the following characters preceded by a dash:

                 l   suppress listing the names of files read

                 o   overlay an existing destfile

                 v   verify that an existing destfile should be overlaid

                 c   inhibit text compression of destfile. The default is
                      to compress the text in destfile.

commandfile   The name of a file containing a list of text files to be
                 read, one per line. The CALLS input is extracted from
                 each of these files and merged into destfile.

destfile   The name of the destination file, to be used as an input
                 file for CALLS. If compression is performed then this
                 file will be of file type 6004.

CALLS

Purpose:　Provides a general-purpose help facility, used either as a help subsystem for other programs, or as the interface to a "database" of information grouped by keywords.

Syntax:　CALLS [*-flags*] [*keyword*]

-flags　A string of characters preceded by a dash (−). Where an argument is required, the next word in the runstring is consumed, delimited by blanks or a comma. The flags are:

C *catalog*　　name of the CALLS catalog to use. By default, /CATALOGS and type extension .CALL are added to the given name. The default catalog is /CATALOGS/CALLS.CALL.

L *listfile*　　divert the text listing to the named file. By default the text is listed to the terminal.

P *pagesize*　　set the number of lines per page for "More..." prompting on the terminal. Default is 22 lines.

B　　build the index file and terminate.

keyword　The keyword for which the associated text is to be listed. If not given, then the default keyword ("[default]") for the selected catalog is listed.

COPYL

Purpose:　Copy one disk LU to another, ignoring file structure.

Syntax:　COPYL [*sourceLu destLu*]

sourceLu　LU of disk to be copied.

destLu　LU of disk to receive data.

Default = interactive mode.

CSYS

Purpose: Copies type 1 (memory image) files from a CS/80 disk to a cartridge tape (CTD) in a memory-based system.

Syntax: CSYS , *filename* , *tapeLu* , *file#* ,
                           [SA : [*nextFile#*]] , [*BCMoffset*]

filename   The name of the type 1 file on disk.

tapeLu   The CTD to be copied to.

file#   The number of the file the VCP will use (this is the "target") to boot the system from the tape.

SA   An optional parameter that specifies that the memory-based system will be hidden inside an ASAVE file. This allows ASAVE files and memory-based systems to reside on the same CTD tape. This option uses a 256-block VCP file for the ASAVE header and an additional 256-block VCP file for the trailer. When using this option, *file#* should not be zero.

nextFile#   An optional subparameter of the SA parameter. It specifies where to put the ASAVE end-of-data record. The default value for *nextFile#* is immediately after the memory-based system.

    To reserve space for more than one memory-based system:

    nextFile# = last file # + {truncation to an integer
                               of ((blk size + 255)/256) }

    last file #   The file # of the last file to be put on tape.

    blk size   The number of disk blocks in the last file on cartridge tape. This value is derived from the information contained in the FMGR or CI DL command. Note that a CTD block is 512 words and a disk block is 128 words.

BCMoffset   The BCM file number. The file # and the BCM offset determine the starting block for the file on tape. For example, if the file # is 1 and the BCM offset is 1, then the file starting block is 288 (256 + 32).

FC

Purpose: Back up and restore files from FMGR file addressing space to magnetic tape or CS/80 cartridge tape.

Syntax: FC [*command*]

command default = interactive mode.

?   List commands and their syntax.

LL   Set list device or file.

TITLE   Set title for tape header file.

CF   Set namr for tape comment file.

ECHO   Turn ON or OFF echoing of commands to list device.

SCRATCH   Specify disk cartridge for scratch files.

COPY   Copy files as specified by parameters.

DEFAULT   Set default source, destination, and options for subsequent COPY commands.

GROUP,   Group multiple COPY commands into a single
EG, AG   COPY operation.

DL   Compile directory list of FC tape.

CL   List FMP cartridge list, or cartridges included on tape.

LC   List comment file from FC tape.

LH   List header file from FC tape.

TR   Transfer to or from FC command file.

EX   Exit FC. If a group COPY is active, it will be completed first.

AB   Abort FC, including any active group COPY.

*   Comment.

FMGR

Purpose:	Manipulate files on FMGR directories. Perform operations that are not available from CI (for example, BL – buffer limits).
Syntax:	FMGR [*input* [*log* [*list* [*severity*]]]] (commands from device)
	FMGR *fi,le,nm* [*severity* [*list*]] (commands from file)
input	LU of input device for FMGR commands. Default is the console log.
log	LU of device used to log and correct diagnosed errors; must be an interactive device.
fi,le,nm	Name of the file containing input to FMGR commands. Specified as 3 words, each consisting of 2 ASCII characters.
list	LU of device used to list results of FMGR commands. Default is LU 6.
severity	Severity code; determines reaction to error. Severity codes and their reactions are:

0 Display error codes; echo commands on log device (default).

1 Display error codes on log device; inhibit command echo.

2 Display error code only if error requires transfer of control to log device for correction (in which case active job terminates); inhibit command echo.

3 Same as 2, except that active job is not terminated when control is transferred to log device.

4 Continue job regardless of error; inhibit command echo; do not transfer to log device.

FMGR Commands

Commands marked with an asterisk (*) can also be used in response to the System> and RTE: prompts. FMGR command parameters must be separated by commas.

AS,*program*,*partition* *

Assign program to partition.

program   A 5-character program name.

partition   A number that identifies the partition that program will be assigned to or reserved for.

Partition = 0 means unassign program from its partition.

BL,*lu*

Examine the buffer limits for an I/O device.

lu   LU number.

BL,*lu*,*buffering*[,*lower*[,*upper*]]

Reset the buffer limits for an I/O device.

lu   LU number.

buffering   BU (buffered) or UN (unbuffered).

lower   Lower buffer limit, ≥ 0.

upper   Upper buffer limit, *lower* ≤ *upper* ≤ available SAM.

BR[,*program*]

Set break bit.

program   5-character program name. (Default = BR primary program scheduled for this terminal or most recent child program.)

The BR command has an effect only if the break bit is tested after the command is entered.

CA,*globalNum* [,*val1* [,*op1* ,*val2* [,*op2* ... ,*opn* ,*val(n+1)*]]]]

Calculate global parameter values.

globalNum   Integers (1 through 9) identifying the global (1G through 9G) to be set to the result of the calculation. The global type can also be specified in the first subparameter. P identifies P-globals; G identifies G-globals. (Default = G-globals.)

val1 – valn Values used in calculations (default = global is nulled).

op1 – opn   Operations performed on operands; can be:

+	adds two operands
–	subtract one operand from another
/	divide one operand by another
*	multiply one operand by another
O[R]	inclusive OR two operands
X[OR]	exclusive OR two operands
A[ND]	AND two operands

CL

List cartridge directory.

CN,*lu,function*[*,pram*4*]

Issue device control request.

lu	LU of device to which control request will be issued.
	LU 1 controls the user's terminal. To control the system console, indicate LU = 100001B.
function	Control function code (0-63B) as defined in the function field of *cntwd* (listed for each driver in the *RTE-A Driver Reference Manual*). A 2-character mnemonic code can be used for some of the more commonly used control functions. The action performed for a particular function code is dependent on the driver. Refer to the *RTE-A Driver Reference Manual* for a complete list of the function codes available for a particular driver.

Mnemonic Code	Equivalent Octal Function Code	*pram*4* Definition	Action
AB	none	None	Abort current request (at head of queue)
AD	24B	Device address	Establish new device address
BF	14B	None	Backward space file
BR	2	None	Backward space record
DP	25B	1-4 parameters	Set device parameters
EO	1	None	Write end-of-file
FF	13B	None	Forward space file
FR	3	None	Forward space record
PR	none	Priority	Establish new device priority
RW	4	None	Rewind cartridge tape
TO	11B	Number of lines on page	Issue top-of-form or line spacing on printer
no mnemonic	0	None	Clear device
no mnemonic	10B	Program name	Enable primary program
no mnemonic	21B	Program name	Disable primary program

pram*4	Optional parameters allowing specification of additional device details as appropriate for a given driver. Specific meanings for these parameters can be found in the *RTE-A Driver Reference Manual* under each driver.

CO,*cartridge1***,***cartridge2*[**,***options*[**,***name1*[**,***name2*[**,***masterSecurity*]]]]

or

CO,*namr1***,***cartridge2*[**,***options*[**,***name1*[**,***name2*[**,***masterSecurity*]]]]

Copy files from one cartridge to another.

cartridge1	Source cartridge identifier (negative LU number or positive CRN).
namr1	File name, and optionally, security code, cartridge identifier (negative LU number or positive CRN), file type, and file size. The *namr* can contain wildcards.
cartridge2	Destination cartridge identifier (negative LU number or positive CRN).
options	Copy options:

C	Clear destination cartridge
D	Dump-mode
E	Eliminate extents
P	Purge source files after
V	Verify

name1	Starting file name.
name2	Ending file name.

masterSecurity

Master security code; two ASCII characters.

CR,*namr*

Create a disk file.

namr	The file descriptor parameter.

CR,*namr,lu,ioMode* **[,***spacing* **[,***fileMark* **[,***dataType***]]]**

Create a nondisk file.

namr   File name and, optionally, security code and CRN; file
     type defaults to 0.

lu   Positive LU number of the nondisk device.

ioMode RD Read only.
     WR Write only.
     BO Both.

spacing Specifies type of spacing for the device.

     BS Backspacing supported.
     FS Forward spacing supported.
     BO Both.

     Defaults: Read-only devices = FS.
          Other devices = no spacing.

fileMark Type of end-of-file mark to be written on the device; if
     omitted, default depends on driver type.

     EO End-of-file mark for magnetic tape.
     PA Page eject for line printer.
     cntrl Control subfunction (equivalent to function
     word code in FCONT call).

dataType Specified type of data on the device.

     BI Binary data.
     AS ASCII data (default).
     cntrl Control subfunction (equivalent to bits 6-10 of
     word IOPTN parameter in OPEN call).

DC,*cartridge*

Dismount logical cartridge.

cartridge Cartridge identifier (negative LU number or positive
     CRN).

DL,,[*masterSecurity*]

List file directory for all mounted file cartridges.

or

DL,*namr*[*,masterSecurity*]

List cartridge and file information on specified file for any directory containing that file.

or

DL,*cartridge*[*,masterSecurity*]

List the directory for the specified cartridge.

cartridge   Cartridge identifier (negative LU number or positive CRN).

masterSecurity
    Code assigned to the system at bootup.

DN,*lu**

Make device unavailable/down.

lu   Device LU number.

DP[*,pram*14*]

Display parameter names or global values.

pram*14   Parameter values or names of global parameters to be displayed; can also be a string of text (default = nothing displayed).

DS,*lu**

Display device status.

lu   LU number of the device.

Device status is indicated by the following codes:

```
                              AV
                  UP          ND
    Response:     DN  DS(xxx) BZ[proga] LK[progb]
                              AB[proga]
                              SY
                              CL

    or:           UNASSIGNED
```

where the displayed codes have the following meanings:

UP   Device considered operable (up) by the system.

DN   Device has been set down as a result of an error reported by the driver or by the DN command.

DS   Eight bits of status posted by the driver at the completion of the last I/O operation. (DVT word 6.) Status displayed in octal format.

AV   Available; no requests pending on the device.

ND   Some other device on the I/O node is busy. No requests can be issued to this device until the other device completes.

BZ   Device is busy processing some read, write, or control request. For a buffered or unbuffered user request, [*proga*] identifies the issuing program by name unless it has gone dormant.

AB   Device is busy processing an abort request.

SY   Device is busy with a system request.

CL   Device is busy with a Class I/O request.

LK   Device has been locked by an LURQ request from [*progb*]. The lock may be shared by cooperating programs.

DU,*namr1*,*namr2*[*,format/control*[*,file*[*,number*]]]

Dump data to a device or existing file. Does not create *namr2*.

namr1   Source of data.

namr2   Destination of data.

format/
control   Record format parameter, end-of-file control parameter, or one of each.

file   A positive integer that specifies the starting point on *namr2*.

number   Specifies the number of files or subfiles to be transferred from *namr1*. Default = 1, unless *namr1* is a disk file and *file* is omitted, then =9999.

EX

Exit FMGR.

GO,*program*,*parameters* *

Resume execution of suspended program.

program   Name of suspended program.

parameters   List of parameters to be passed to the program only if
the program has suspended itself.

IF,*val1*,*operator*,*val2*[,*skip*]

Conditional skip.

val1, *val2*   Values to be compared; one or both can be global
parameters.

operator   Relative operator used to compare values of *val1* and
val2.

Keyword	Operation
EQ	*val1* = *val2*
NE	*val1* ≠ *val2*
LT	*val1* < *val2*
GT	*val1* > *val2*
GE	*val1* ≥ *val2*
LE	*val1* ≤ *val2*

skip   Number of commands to skip when relation between
val1 and *val2* is true.

IN,[*masterSecurity* [*,cartridge,crn,label* [*,firstTrack* [*,dirTracks*]]]]

Initialize a logical cartridge or change the description of an initialized cartridge.

masterSecurity

    Security code specified at bootup.

cartridge +CRN or −LU

crn   Cartridge reference number or name of cartridge to be initialized. 0<*crn*<32768, or two ASCII characters.

label   ASCII identifier assigned to cartridge: up to six ASCII characters specified exactly like an FMP file name.

firstTrack First FMP track on cartridge; (default = track 0 assumed).

dirTracks Number of tracks to be allocated for the file directory (default = 1).

IN,*masterSecurity* − −*newSecurity*

Change the master security code of the file system.

masterSecurity and *newSecurity* are separated by two minus signs.

masterSecurity

    Security code specified at system bootup.

newSecurity

    New master security code.

IO[,*luList*]

Display system I/O configuration on the list device.

lulist   Up to seven LU numbers separated by commas.

IT,*program* [,*resCode* ,*mult* [,*hr* [,*min* [,*sec* [,*ms*]]]]]

or

IT,*program* [,*resCode* ,*mult* [,*-offset*]]

Display or set the initial execution time and the rescheduling interval of a program.

program	Program name.
resCode	Resolution code:

> MS = tens of milliseconds
> SC = seconds
> MN = minutes
> HR = hours
> OF = remove from time list

mult	Multiplier for resolution code.

hr,min,sec,ms

> Initial execution time.

-offset	Initial wait factor (depends on resCode).

Rescheduling interval = *resCode* * *mult*

Initial wait = *resCode* * *offset*

LI,*namr* [,*format* [,*line1* [,*line2*]]]

List contents of a file or LU.

format	Specifies list format:

> null   ASCII source format
> A   ASCII source format
> B   binary format
> D   directory information only
> S   ASCII source format

line1	First record to list (default = 1).
line2	Last record to list.

LL [,*namr*]

Display/change list device.

MC,*lu* [*,lastTrack*]

Mount logical cartridge.

> *lastTrack*   Last track on the cartridge available to the file
> management system (default = the last track defined
> for the disk LU at system generation time).

OF[,*program* [*,severity*]]*

Terminate/abort program.

> *program*   Program name. (Default = OF primary program
> scheduled for this terminal or most recent child.)

> *severity*   FL   (flush output (default)),
> ID   (flush output and release ID segment), or
> DR   (release ID segment if dormant).

ON,*program* [*,parameters*]

Schedule program for timed execution.

> *program*   Name of program in time list.

> *parameters*   Starting parameters for the program.

This command acts only on programs that have been time-scheduled
with the IT command.

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

Suspend execution of a transfer file.

> *lu*   Default = the log device.

> *message*   Message to be displayed on *lu*.

PK[,*cartridge*]

Pack logical cartridge.

PL[,*status*]

List programs and status.

status   2-character status code, or MB, IT, or PT (default = list
              of all programs, along with status information for
              each).

MB   List memory bounds of each program.

IT   List all programs in the time list; display
        information:

        R = resolution code
        M = multiple

PT   List all partitions sequentially, along with
        lengths and occupants of each.

Default display information (when PL is specified with no parameter):

First column:

PR   = Priority.

Second column:

PC   = Program counter, the octal address of the last
          instruction executed.

Memory status (third column):

M   = Program in memory.
T   = Time-scheduled program.
L   = Program being loaded from disk.
S   = Program being swapped to disk.
blank = none of the above.

Status information (fourth column):

AB   Program in the process of being aborted.

BL (*lu*)

        = Program requested buffered output or Class
          I/O to the given LU, but the device already
          has requests exceeding its allotment of SAM.

CL   = Either the program did a Class I/O Get
          (EXEC 21 call) and the queue for the given
          class number is empty, or the program
          attempted to allocate a class number and
          none was available.

DN (*lu*)

> = Program suspended because it made an I/O request to the given down LU.

IO (*lu*)

> = Program waiting for an I/O request to complete.

LD = *proga* made an EXEC 28 or EXEC 8 request to restore a real-time program or segment from the disk.

LK (*lu*)

> = Program suspended because it made an I/O request to the given LU, which is locked to another program.

MM = Program waiting for memory to become available.

OF = Program dormant.

PA = Program suspended itself with a pause statement or an EXEC 7 call.

QU (*progb*)

> = *proga* attempted to schedule *progb* with queue (EXEC 23 or 24 call) but *progb* was already active.

RN (*progb*)(−GLOB)(−NONE)

> = Program waiting for *progb* to release a resource number. −GLOB indicates that the resource number was locked globally. −NONE indicates the program attempted to allocate a resource number and none was available.

SC = Program scheduled (waiting for CPU time) and will execute when all higher programs are suspended or dormant.

SS = Operator suspended the program with the SS command.

SR = Program suspended waiting for another program to exit a shared subroutine.

TM = Program requested a timed delay of its own execution (EXEC 12 call, *name* = 0).

WT (*progb*)

    = *proga* has scheduled *progb* with wait (EXEC 9 or 23 call) and will continue execution when *progb* completes.

XQ   = Program executing.

PR,*program*,*priority* *

Change program priority.

program   Program name.

priority   New priority number.

PS[,*program*] *

Display program status.

program   Name or ID segment number of program for which status is desired (default = currently executing program).

Default display information and status codes are the same as those for the PL command.

PU,*namr*[,*masterSecurity*]

Remove selected files and their extents from a cartridge.

masterSecurity

    Master security code specified at bootup.

RN,*oldnamr*,*newname*

Rename a file.

oldnamr   Old file name[:*sc*[:*cr*]]

newname   New file name to replace existing file name.

RP,*filenamr*[,*newname*]

Set up an additional ID segment (restore) for a type 6 program file for execution.

filenamr   File descriptor of the type 6 program file to be restored.

newname   Additional program name to be used.

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

Schedule a program for immediate execution, and wait for completion.

program   5-character file descriptor.

parameters   Parameters to be passed to the program.

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

Set global parameters.

p1 through *p9*

        Values assigned to global parameters 1G through 9G.

SS,*program* *

Suspend active program.

program   Name of active program.

ST,*namr1*,*namr2*[,*format/control*[,*file*[,*number*]]]

Store data in a device or new file. Creates *namr2* if it is to be a disk file.

namr1 Source of data; can be a disk file descriptor or an LU number (positive).

namr2 Destination of data; can be a disk file descriptor or an LU number (positive).

format/control

     A record format parameter, and end-of-file control parameter, or one of each. Record format parameters:

     AS ASCII records.
     BR Binary relocatable records.
     BN Binary relocatable records; no checksum.
     BA Binary absolute records.

     EOF control parameters:

     IH No file marks are passed to *namr2*.
     SA All file marks are passed to *namr2*.

     If the EOF control parameter is omitted, embedded file marks are not passed to *namr2*, but the terminating file mark is.

file   The starting point on *namr2*.

number The number of files or subfiles to be transferred from *namr1*.

SV[,*severity*[,*global*][,IH]]

Display/set severity code.

severity New severity code; it can be:

0   display error codes and echo commands on log device (default).

1   display error codes on log device; no command echo.

2   display error code only if error requires transfer of control to log device for correction, no command echo.

3   same as 2.

4   continue command file automatically if a FMGR command error is encountered; no command echo, no transfer to log device, and no command file abort occurs.

100x   if the above severity codes are added to 1000, when an error occurs the error is expanded as if ?? was entered.

global Optional global number in the range 1-9, into which the severity code is placed.

IH Inhibit echo of command entry.

SZ,*program* *

Display program size information.

program   A 5-character program name.

The information is displayed as:

last=*lllll* min part=*pppp* EMA/

WS=*eeee* mseg=*mm* VMA=*vvvvv*

lllll   = the address of the last word plus 1 of program. If the
             program is segmented, *lllll* is the address of the last
             word plus 1 of the largest segment.

pppp   = minimum required partition size. If the program is of
             the EMA type, *pppp* equals the program size plus the
             EMA size plus 1. If the program uses shareable EMA,
             pppp equals the program size.

eeee   = EMA size in pages (EMA programs only).

mm   = MSEG size (EMA/VMA programs only).

vvvvv   = VMA size in pages (VMA programs only).

SZ,*program*,*progSize* [,*msegSize*] *

Modify program size.

program   A 5-character program name whose size is to be
               modified.

progSize   New required program size in pages for non-VMA
               programs or the new EMA size for EMA programs, not
               including PTE.

               Range is 2 ≤ *progSize* ≤ 1022 for EMA size.

msegSize   New MSEG size for EMA programs.

               Range is 1 ≤ *msegSize* ≤ 30.

TM

Display current system time, in the format:

hr:*min*:*sec mo date*,   *year*   *day*

TM[,*hr*[,*min*[,*sec*[,*mo*[,*date*[,*year*]]]]]]

Set system clock.

hr	hour, 0 to 23.
min	minute, 0 to 59.
sec	second, 0 to 59.
mo	month, 1 to 12.
date	day of the month, 1 to 31.
year	year, 1976 to 2144.

After the system clock is reset, the system responds with the current system time.

TO,lu

Display device timeout limit, in the format:

TO # *lu* = *intervals*

intervals   Number of 10-ms intervals used as the timeout value for device LU.

If *intervals* = 0, timeout limits do not apply to this device.

TO,*lu*[,*intervals*]

Modify device timeout limit.

intervals   Number of 10-ms intervals to be used as the timeout value for device LU; $0 \leq intervals \leq 65534$.

If *intervals* = 0, timeout limits do not apply to this device.

After the timeout is reset, the system responds with the current device timeout.

TR[,,*parameters*]

or

TR,*namr*[,*parameters*]

or

TR,−*integer*[,*parameters*]

Transfer control to a file or device.

namr	A file or LU to which control is transferred; pointers to the namr are maintained in a transfer stack and can be nested up to 8 levels deep.
−integer	A negative integer; control is transferred back the specified number of levels in the transfer stack; if the integer is greater than the current level, the stack is reset to the top (default = −1; that is, TR is the same as TR, −1).
parameters	Values to be set for the global parameters 1G through 9G; position determines the global parameter to which the value is passed; omitted global parameters are unchanged.
	A colon (:) or comma (,) can replace TR as the command code.

UL,*label* *

Unlock shareable EMA partition.

label	A 6-character label name that identifies a shareable EMA partition label.

UP,*lu* *

Make device available (up).

VS,program *

Display VMA size information.

program   A 5-character program name.

The information is displayed as:

`last=`*lllll* `min part=`*pppp* `EMA/`

`WS=`*eeee* `mseg=`*mm* `VMA=`*vvvvv*

lllll   = the address of the last word plus 1 of program. If the program is segmented, *lllll* is the address of the last word plus 1 of the largest segment.

pppp   = minimum required partition size, equal to the program size plus the working set size.

eeee   = working set size in pages (not including PTE).

mm   = MSEG size.

vvvvv   = the program's VMA size in pages.

VS,*program,vmaSize* *

Modify VMA size.

program   A 5-character program name whose VMA size is to be modified.

vmaSize   New VMA size in pages.
Range is $32 \leq vmaSize \leq 65536$.
Note: 0 or ≥ 64512 is construed as 65536.

WS,*program* *

Display VMA working set size information.

program   A 5-character program name for which information is desired.

The information is displayed in the same format as the VS command.

WS,*program***,***wsSize* *****

Modify VMA working set size.

program A 5-character program name whose VMA size is to be modified.

wsSize New working set size in pages (not including PTE). Range is 2 ≤ *wsSize* ≤ 1022.

XQ,*program***,***parameters* *****

Schedule a program for immediate execution, and do not wait for completion.

program A 5-character program name,
      or
     a file descriptor that identifies a type 6 program file to be executed; cannot be an LU number.

parameters Parameters to be passed to the program.

??[,*error***]**

or

??[,AL]

Explain FMGR error.

error FMGR error number (default = most recent FMGR error).

AL   All error code explanations are listed to the list device.

FORMC

Purpose:	Verify integrity of CS/80 disk and tape drives, spare a disk block, or format a disk volume or tape.
Syntax:	FORMC [*listLu*] [*command*] *lu* [*params*]
listLu	LU of list device.
command	One of the following, or its first two letters (default = interactive mode):

?   List FORMC commands.

END | EXIT | / EXIT
    End FORMC.

ABORT | / ABORT
    Abort FORMC.

VERIFY *mediaLu* [*start number*]
    Verify integrity of CS/80 disk or tape.

mediaLu	LU of CS/80 disk or tape to be verified.
start	Number of disk track or tape block at which verify is to begin. Default = entire disk or tape.
number	Decimal number of disk tracks or tape blocks to be verified.

SPARE *mediaLu* *track*
    Spare blocks on specified track.

mediaLu	LU of CS/80 disk or CTD (cache tracks) to be spared.
track	Number of track containing blocks to be spared.

FORMAT *mediaLu* [*interleave*]
    Format specified disk or tape.

mediaLu	LU of CS/80 disk or tape to be formatted.
interleave	Disk sector interleave factor. Decimal number between 1 and 32 (default = 1).
params	Same as the preceding commands.

FORMF

Purpose: Verify integrity of a disk or format a disk.

Syntax: FORMF [*logLu* [*command lu* [*interleave*]]]

  logLu   LU of log device.

  command   VE (verify disk integrity) or FO (format disk).

  lu   LU of disk.

  interleave   Disk sector interleave factor.

Exit FORMF with EX, EN, or /E; abort it with AB or /A.

FORMT

Purpose: Format a floppy disk, initialize a hard disk, spare a track on a hard disk, verify data on a disk LU, or reformat a hard disk.

Syntax: FORMT [*logLu*] [*command diskLu* [*pram*]]

  logLu   LU of device from which FORMT queries are answered. Default is user terminal.

  command   One of the following:

FO	Format floppy disk.
IN	Initialize hard disk.
SP	Spare track on hard disk.
VE	Verify data on disk LU (read-only verify).
RE	Reformat hard disk.

  diskLu   LU of disk.

  pram   For command = FO, *pram* = sector fill number; for command = SP, *pram* = track to spare; for other commands, *pram* is empty.

FOWN

Purpose: Displays owners of and disk space used by files specified.

Syntax: FOWN [*options*] [*fileMask*] [*fileMask2* ...]

options One or more of the following (note, however, that the −k and −m options are mutually exclusive):

-k report disk usage in Kbytes. Changes value returned in $RETURN_S.

-m report disk usage in Mbytes. Changes value returned in $RETURN_S.

-q (quiet) inhibit error reporting.

-- marks the end of the options. Only required when the first mask begins with a hyphen (−).

fileMask Specifies the files to scan. The default mask is
fileMask2 ... /@.@.sgl, which displays information about all FMP files on the system. Symbolic links to directories are not followed.

FPACK

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

Syntax: FPACK *lu*

lu LU of volume to be packed.

FPUT

Purpose: Install bootable systems and diagnostics in space reserved by IN command.

Syntax: FPUT *file disklu* [*VCPfilenumber*] [*BCMoffset*]

file File descriptor of file to be copied to the bootable area on the CI file system volume given by *disklu*.

disklu CI file system volume to which specified file is to be copied.

VCPfilenumber
Offset within the bootable area in 256-block units.

BCMoffset Offset within the bootable area in 32-block units.

FREES

Purpose:	Report total free space and size of largest free space on CI file system volume.
Syntax:	FREES [*options*] [*DiskLu*] [*DiskLu* : *DiskLu*] ...
options	One or more of the following:

-v   Inhibit the inverse video bar graph.

+l:*lu* Send the listing to the given LU.

-g   Bar graphs are not relative to the largest disk.

disksz where *disksz* is one of the following:

    +t   Report disk size in number of tracks.
    +m   Report disk size in Mbytes.

sort where *sort* is one of the following:

    -s   Inhibit sort, report in cartridge list order.
    +h   Sort by largest hole size.
    +d   Sort by disk size.

quiet where *quiet* indicates quiet mode and is one of the following:

    +q   return all free space information to the scheduling program in $return_s.
    +qd   return only disk size in $return_s.
    +qf   return only free size in $return_s.
    +qh   return only largest hole size in $return_s.
    +qr   return only reserved size in $return_s.
    +q%f   return only percent free in $return_s.
    +q%m   return only percent max in $return_s.

DiskLu   Specifies a disk LU to display; it can be repeated.

DiskLu : *DiskLu*

    Specifies a range of disk LUs to display; if no disk LU is specified, all CI volumes are listed.

FSCON

Purpose:	Convert FMGR cartridge to CI (hierarchical) structure.
Syntax:	FSCON *lu*
lu	LU of FMGR cartridge to be converted.

FST

Purpose: Back up and restore files from the file addressing space to magnetic tape, CS/80 cartridge tape, DDS media, or an archive file on disk.

Syntax: FST [*command* | *command* | ...]

command   Default = interactive mode.

Information Commands:

HE|?[*cmd*]   List commands and options and their syntax.

SH   Show option settings and selection status.

Backup and Restore Commands:

BA *mask* [*destMask*] [*secCode*]
      Select files to back up.

DF *fileDesc*   Specify directory file name, location or size.

GO   Begin executing backup/restore.

RE [*mask*] [*destMask*] [GR|EG|AG]
      Select files to restore.

SC [*filename*]   Select tape's comment file (backup only).

TA [ON|OFF|A|B] [C]
      Use UNIX TAR archive format.

TI *title*   Assign a title to the archive (backup only).

UN [*mask*]   Remove selected files from backup or restore.

Listing Commands:

DL [*mask*]   List the archive's directory file.

LC   List the archive's comment file.

LH   List the archive's header.

LI [*mask*]   List the directory file from the selection.

LL *device*|*file* [a] [o]
      Select a log device/file.

LN [*mask*]   List the unselected files (restore only).

Tape Related Commands:

MT [*tapeLu*|*file*] Assign a tape LU or archive file name.

NE [*append#*]   Position to the next append of data.

PO [*append#*]   Position to a specific append of data.

PR [*append#*]   Position to the previous append of data.

| SD *[density]* | Set the tape density for saving. |
| SE | Lock tape LU and check tape status or open archive and check the file's status. |

Miscellaneous Commands:

EM *#pages*	Modify the SHEMA partition size.
EX	Exit FST.
TR *filename*	Transfer control to a command file.
RU *prog*	Run a program.
*	Comment.
/ *[n]*	Display the FST command stack.

FST Options

Options may be specified in the runstring or interactively as follows:

FST> *opt* *[opt]* ... [ON|OFF] *[opt]* ... [ON|OFF]

Append	Append this backup to the data already on the tape.
Brief	Only show errors and status messages.
Clear	Clear the disk file's backup bits.
Duplicate	Replace duplicate files.
Faulty	Restores files from a partially overwritten tape.
Inhibit	Inhibit the tape rewind between backups.
Keep	Keep tape online when backup/restore is complete.
Lock	Lock any disk LUs used.
MinDir	Minimize the FST directory file size during restores.
Normal	Backup symbolic links as normal files (follow links).
Original	Restore files to their original main size.
Purge	Purge the disk files after backing up the files.
Quiet	Report messages only to the log device/file.
RwndOff	Rewind and go offline at exit.
SrchApp	Search through appends during RESTORE.
Update	Replace duplicate file if file has been updated.
Verify	Verify the files during the backup/restore.
Whole	Back up all the blocks reserved for the file.
Yes	Write over the tape without asking.
Z	Pause during restore phase for disk full errors.

FVERI

Purpose: Verify that data within a hierarchical file system volume is consistent. Fix some of the inconsistencies found by the verification.

Syntax: FVERI [*lu* | *mask*] [*options*]

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

mask   Mask describing a directory in which all files and/or subdirectories are to be verified.

options   One or more of the following, in any order:

+L *file* | *lu*   List file or device for errors.

+B   Verify bit map only; illegal if a mask is also specified.

+FB   Fix the bit map.

+FD   Fix illegal directory entries (set purged).

+FF   Fix file directory information.

+OK   Do fixes without asking each time.

??   Online help.

INSTL

Purpose: Initialize boot extension (BOOTEX) file for disk LU.

Syntax: INSTL [*snap*] [*system*] *bootDest lu*
                                    [*bootSource*] [*option*]

snap   Snapshot file of target system. Default = same as host
             system.

system   System file of target system. Default = same as host
             system.

bootDest   File to contain new boot extension (must be in target
             system), or number that is an offset into reserved space
             on *lu*.

lu   Disk LU of boot extension file (on target system).

bootSource Name of boot extension file on host system. Default is
             bootDest, if it is a file.

option   N = No console.
             E = Enable CS/80 timeout retry.

KILLSES

Purpose: Terminate a session immediately, killing all programs
             associated with session.

Syntax: KILLSES [*sessnumber*] [OK]

sessnumber Specifies the number of session to be killed. If not
             specified, user is prompted for number of session to be
             killed.

OK   Executes immediately without prompting user to verify.

MERGE

Purpose:	Combine two or more input files into a single output file.
Syntax:	MERGE [*-options*] [*fromfile fromfile . . .*] [*destfile*]
fromfile	One of the following:

- File containing list of names of files to be merged.
- File to be merged into *destfile*.
- Mask of files to be merged.

destfile	File to receive merged files.
options	Any of following characters preceded by dash (−).

D	Remove Debug records.
L	Do not list file names merged.
O	Overlay existing files without asking.
V	Verify that existing files may be overlaid.
Z	Suppress zero-length records between files.

METER

Purpose:	Display information about system and user processes.
Syntax:	METER [*lu*] [*commands*]

Once running, METER recognizes the following single-key commands (enter uppercase for a primary sort key and lowercase for a secondary sort key):

A	Display all processes
D	Sort in decreasing order
E	Exit
H	Sort by Histogram
I	Sort in increasing order
L	Sort by LU (session number)
N	Sort by name
O	Sort by Owner
P	Sort by priority (default)
S	Sort by state
T	Sort by total CPU
U	Display only user processes
?	Display help screen
space	Exit

MPACK

Purpose: Compact hierarchical files and pack hierarchical file volumes.

Syntax: [RU,]MPACK [options] mask|lu [options]

options   MPACK has compacting, reporting, packing, and logging options. Precede options with "+". Enter options anywhere in runstring but enter them only once. Compacting takes place before packing. If you use the P or V option, you must specify an LU number.

Compacting/Reporting Options:

+R   Remove extents from files.

+T[n[%]]   Truncate all wasted blocks from files, or truncate all but *n* or *n*% of wasted blocks.

+C[n[%]]   Remove all extents and truncate all wasted blocks or truncate all but *n* or *n*% of wasted blocks; +C cannot be used in same runstring with either +T or +R.

+B   Adjust file size to blocks per bit value. Can be used with +R, +T, or +C.

+E*n*   Select files with *n* or more extents.

+W*n*   Select files with *n* or more wasted blocks.

+W*n*%   Select files with *n*% or more wasted blocks.

+A   AND the +E and +W qualifiers.

+D   Include directories in the compacting.

+Q   Quiet mode. Suppress individual file information and only report totals.

Packing Options:

+P   Pack disk LU.

+OK   OK to overlay data during pack.

+V   Enable visual mode; display bit map.

Logging Option:

+L *file*   Log output to a file or device.

NOTIFY

Purpose:	Send a one-line message to another user's terminal. Notification may be turned on or off for your session. If notification is turned off, messages sent to your session are thrown away without being printed.
Syntax:	NOTIFY [-a] *user message* NOTIFY ON NOTIFY OFF
-a	Inhibits notifying alias logons; treats *user* as a logon name only.
user	Specifies who is to receive the message, and may name either a logon name, a session number, an alias name, or the name ALL in which case the message is sent to all sessions logged on.
message	One-line message to be sent by the NOTIFY utility.

OLDRE

Purpose:	Change extended relocatable record formats to nonextendable formats.
Syntax:	OLDRE *namr*
namr	Name of type 5 file. Begins with %.

PRINT

Purpose:	Print one or more files. If entered without parameters, the status of jobs in the printer queue is displayed.
Syntax:	PRINT [*file1*[,*file2*[, ...]]] [*lu*] [*options*]
file1...	Name of file or file mask for files to be spooled for printing.
lu	LU of printer. Default = 6.
options	One of the following;

+A:*file*	Append output to the specified file.
+B:*string*	Print specified string as banner headline.
-B	Do not print default banner.
+C:ON|OFF	Set carriage control option on or off.
+F:*number*	Advance paper the specified number of pages after all files are printed.
+H:ON|OFF	Print (ON) or do not print (OFF) the header before each file.
+I:*number*	Indent the specified number of pages.
+L:*number*	Specify number of characters per line in banner.
+M:*number*	Merge successive files with specified number of blank lines between them.
+N	Produce a numbered listing for each file in the runstring.
+O:*file*	Send each file listed in the runstring to the specified file.
+P[:OK]	Purge printed files that have been sent to the printer.
+Q	Suppress file mask verification.
+S	Suppress "Print job supervised by PRINX".

+T:*file*	Transfer to *file* to get the list of files to print.
+W:*dir*	Search specified directory, not current directory or files.
+X:*number*	Print all files contained in runstring the specified number of times.
+Y:*number*	Specify the number of spaces between characters in the banner.
+Z	PRINT waits for PRIN0 to finish before completing.
+?	Return a short explanation of available options.

RINFO

Purpose: Reset multiuser accounting information in the user configuration file. (Applicable only for systems not using groups).

Syntax: RINFO [*UserMask1* [*UserMask2* [...]]]

UserMask   Name or file mask for the users whose accounting information is to be set to 0. Default = current user.

SAM

Purpose: Print information about the status of System Available Memory (SAM). If system contains Extended System Available Memory then SAM will report its status also.

Syntax: SAM [AL] [XS] [*lu*|QU]

AL   A complete listing of all system memory blocks and a summary of the current status of system memory is printed. If omitted, only the summary is printed.

XS   Return information about XSAM in the return parameters.

lu|QU   LU to direct output to; default is LU 1. QU, quiet mode; output goes to bitbucket (LU 0).

SCOM

Purpose:	Compares two input files and identifies their differences.
Syntax:	SCOM *file1 file2* [[+ | ~] *listfl*] [*options*] [*rematchlns*] [*maxchars*] [*difflimit*]
file1	Input files to be compared.
file2	
listfl	Output file/device to receive results of comparison. *Listfl* must be new file; +*listfl* appends to existing file, ~*listfl* overlays an existing file. Terminal is default.

options

F1	list only lines unique to *file1*
F2	list only lines unique to *file2*
BO	list lines common to both files
NN	suppress line numbering
TB	include trailing blanks as factor in determining line match
Dx	use *x* as "don't care" character
Cx	ignore blank lines with *x* in column 1
NH	no heading is output
NT	no tail is output
IB	trailing blanks are insignificant
IT	ignore EDIT/1000 time stamps
IC	ignore compile times in relocatable records for binary record compares
ET	create EDIT/1000 transfer file to convert *file1* to *file2*
ER	same as ET with ER command added to end of transfer file
BR	binary record-by-record compare
BB	binary block-by-block compare
TC	force text compare on binary file

Options must be specified without intervening commas, for example, "NNF1F2C*".

Default is F1F2. Default options remain in effect if other options are specified. However, if one default option is specified in the runstring, only that option is enabled with the selected options; the other default option is suppressed.

rematchlns	Specifies number of lines that must match in files before a mismatch is ended. Default is three lines.
maxchars	Specifies maximum record size for input files. Default is 256 characters.
difflimit	Maximum number of differences to report.

SESLU

Purpose:	Modifies and lists session bit maps.
Syntax:	SESLU [+S:*sess*] [[-]*lu* [:*lu*]] [[-]*lu* [:*lu*]]
	. . .
+S:*sess*	Specifies the number of session whose LU Bit Map is to be modified or listed. If not specified, SESLU defaults to caller's session.
[-]*lu* [:*lu*]	If positive, specifies LU number or range of LUs to set in session bit map.
	If negative, specifies LU number or range of LUs to clear in session bit map.
	If not supplied, LU bit map is listed.

SINFO

Purpose:	Display multiuser accounting information in the user configuration. (Applicable ONLY for systems NOT using groups).
Syntax:	SINFO [*UserMask1* [*UserMask2* [. . .]]]
UserMask	Name or file mask for users whose accounting information is to be displayed. Default = current user.

SPORT

Purpose:	Displays the status of serial ports using DDC00 in RTE-A or DVC00/DV800 in RTE-6/VM.
Syntax:	SPORT [*port_lu* [*display_lu* [*waitflag*]]]
port_lu	Serial port LU for which current status is requested.
display_lu	LU to which the output is directed. Output can be directed to another LU, but not to a file.
	Both LUs default to the session LU.
waitflag	Specifies a non-zero value to force a "wait" for a busy port; in other words, you can have SPORT "hang" on the port until the current request completes.

TF

Purpose:	Back up and restore files from the file addressing space to magnetic tape, CS/80 cartridge tape, or DDS media.
Syntax:	TF [*command*] TF TR *commandfile*

command   Default = interactive mode.

?	List commands and their syntax.
COPY	Copy files as specified by parameters.
TITLE	Set title for tape header file.
DEFAULT	Set default source, destination, and options for subsequent COPY commands.
GROUP, EG, AG	Group multiple COPY commands into a single COPY operation (start GROUP, End Group, Abort Group).
LL	Set list device or file for LH or DL.
LH	List header file from TF tape.
DL	Compile directory list of TF tape.
TR	Transfer to or return from TF.
EX	Exit TF.
*	Comment.

7

LP Spooler

accept

Purpose:　　Allows print requests to be submitted to the named destinations.

Syntax:　　　`accept` *destination* [*destination* ...]

　destination　Name of a printer or class to which you want print requests to be submitted.

You must be a superuser to run this program.

Returns:

$RETURN1 = the number of errors encountered; zero if none.

Error Messages

`accept: destination` *"dest"* `was already accepting requests`

The named destination already accepted new submissions.

`accept: destination` *"dest"* `non-existent`

An invalid destination was specified.

cancel

Purpose: Removes requests from the LP spool system.

Syntax: cancel *id|dest* [*id|dest*...]
                     [-a | -e | -u*user*[*@host*]] [-i]

id   Request ID, as reported by lpstat, for a request to
               cancel. You may cancel a request that is owned by
               another user only if you are a superuser.

dest   Name of destination printer or class for which −a, −e,
               or −u options apply. If none of these options are
               given, each destination specified must be the name of a
               printer; the request currently printing on each named
               printer will be cancelled. If dashed options are given,
               the options are applied to each destination specified.
               You may cancel a currently-printing request that is
               owned by another user only if you are a superuser.

-a   Cancels all of your requests for each *dest* specified.

-e   Empties queue for named destinations. All requests
               bound for those destinations are cancelled. You must
               be a superuser to use this option.

-u*user*[*@host*]
               Cancels all requests for each *dest* specified owned by
               user. If user name is given in format *"user@host"*, all
               requests owned by *user* that originated on host *host* are
               cancelled, otherwise only requests that originated on
               the local host are cancelled. More than one −u option
               may be given to cancel requests for more than one
               user. You must be a superuser to use this option.

-i   Inhibits remote cancels; cancel only local requests
               destined for remote printers. If this option is not
               given, remote cancel requests will be sent to remote
               printers, as discussed below.

If a *dest* parameter, or a destination within an *id* parameter, names a
remote printer, both local and remote cancellation may be performed.
Requests on the local system that meet the criteria are cancelled; the
requests are queued waiting to be transferred to the remote host. If
the −i option is not given, the remote host is asked to cancel requests
on the remote system that meet the criteria.

Returns:

   $RETURN1 = the number of errors encountered; zero for none.

   $RETURN2 = the number of requests cancelled locally.

Error Messages

`cancel: must be superuser to use "-e"`

You must be a superuser to use the −e option.

`cancel: must be superuser to use "-u"`

You must be a superuser to use the −u option.

`cancel: unknown option: parm`

An unrecognized option was given.

`cancel: "dest" does not name a legal destination`

The destination name entered is unknown. Enter "`lpstat -a`" to display a list of legal destinations.

`cancel: you do not own request requestid`

You do not own the named request. You must be a superuser to cancel another user's request.

`cancel: request "requestid" not found`

The named request was not found. Enter "`lpstat -o`" for a list of requests.

`cancel: printer "printer" was not busy [locally]`

`cancel` was told to cancel the request active on a printer, but the named printer was not busy with a request. If the printer is a remote printer, then the word "locally" appears at the end to inform you that this message applies only to the local operation of the printer. If the −i option was not given, `cancel` will attempt to perform the cancel operation on the remote host also.

`cancel: no requests selected on "dest" [locally]`

No requests were selected for the −e or −u options. If the printer is a remote printer, then the word "locally" appears at the end to inform you that this message applies only to the local operation of the printer. If the −i option was not given, `cancel` attempts to perform the cancel operation on the remote host also.

`cancel: warning: fmp error /usr/spool/lp/hostname`

An FMP error was encountered when retrieving the local host name from the named file.

disable

Purpose: Disables the named printers, preventing them from printing requests sent to the printers or to classes of which the printers are members.

Syntax: disable [-c] [-r["*reason*"]] *printer* [*printer* ...]

-c   Cancels request if a request is currently active on printer being disabled. By default, active requests remain in queue to be printed on printer once it is enabled, or to be printed on another member of the destination class.

-r["*reason*"]

Specifies reason for taking printer out of service. This reason will be reported by lpstat. Must be less than 80 characters; should be surrounded by double quotes. From CI, also use backward quotes to prevent upshifting and comma-insertion of the reason. The specified reason will apply to all printers named after the −r option; more than one −r may be specified, intermixed with printer names, to associate different reasons with different printers. If not specified, "reason unknown" is used.

The enable and disable programs affect only the local spool system. Only the local operation of a remote printer is modified. A disabled remote printer will not transfer requests to the remote host, but the operation of the printer on the remote host is unaffected.

Returns:

$RETURN1 = the number of errors encountered; zero if none.

Error Messages

disable: printer *"printer"* was already disabled

The named printer was already disabled.

disable: missing trailing quote: *parm*

A leading quote was found for the "reason" string, but no matching trailing quote was found.

enable

Purpose: Allows a printer to print requests queued for that printer, or for a class of which the printer is a member.

Syntax: enable *printer* [*printer* . . .]

Each named printer is enabled. New printers added to the LP spooler are initially disabled.

The enable and disable programs affect only the local spool system. Only the local operation of a remote printer is modified. A disabled remote printer will not transfer requests to the remote host, but the operation of the printer on the remote host is unaffected.

Returns:

$RETURN1 = the number of errors encountered; zero if none.

Error Messages

enable: printer *"printer"* was already enabled

The named printer was already enabled.

lp

Purpose: Creates a "request" that the LP spool system print the files specified.

Syntax: `lp [-opts] file|mask [file|mask...] [<fileoffiles]`

opts   One or more options preceded by a dash (−), each separated by a space or comma, specified in any order before first file or mask is given. Valid options are described below.

file | mask   File or mask of files to be printed.

<fileoffiles   File that contains a list of files to print, one per line.

Options

−c   Copy files to be printed into the spool request directories (directories under `/usr/spool/lp/request` that the spool system uses to access the files in a request); do not attempt to create symbolic links to the files. By default, lp tries to create symbolic link files in the request directories to the files to be printed. If symbolic links are created, the files must not be opened exclusively at the time the spool system actually prints the files. Any changes made to the files since the files were spooled will be reflected in the printed output. Files are always copied on systems that do not have symbolic link capability. Files that name numeric LUs are always copied.

−d*dest*   Print files using named destination, either printer name or name of a class. If a destination is not specified, the destination named by environment variable `$LPDEST` is used. If that variable does not exist, the system default destination is used (set by the lpadmin program).

−m   Send mail when the request is printed.

−n*copies*   Print the given number of copies, an integer from 1 to 32767. The default is 1.

−o*option*   Specify an interface-specific option to be passed to the printer interface used to print the request. More than one −o option may be given to pass more than one option. See the documentation on the destination printer interface program or routine for the options that it supports.

Because RTE case folds the lp runstring by default, and because most options are lowercase, lp will reverse the upper or lowercase of each character in the option. For example, if a printer interface accepts option "BSDp", the lp runstring from CI might be:

CI> lp -o`bsd`p *filename*

-p*priority*   Specify the priority of the request, an integer from 0 to 7. If not specified, the default priority for the destination printer is used (set by lpadmin), or the highest default priority of all printers that are a member of the destination class is used.

-s   Suppress informational messages.

-t *"title"*   Use *title* on banner page of output, must be less than 80 characters and must be surrounded by double quotes if it contains blanks or commas. When executing lp from CI, use backward quotes to prevent upshifting and comma-insertion of the title.

-w   Write message on your terminal when request is printed.

Returns:

$RETURN1 = the number of errors encountered; zero if none.
$RETURN_S = the request ID assigned, if successful.

Error Messages

```
lp: no files to print specified
```

No files to print for the request were given in the runstring.

```
lp: you must enter a destination; no system
default
```

No destination for the request was given (using the −d option), and you do not have a default destination set in the $LPDEST environment variable, and no system default destination has been set. The system default destination may be set using lpadmin.

```
lp: Destination "dest" is not accepting submissions;
the reason is:
```
      reason not accepting requests

The destination rejects new submissions. The second line gives the reason set by the system manager when the destination was made to reject submissions. The destination can be made to accept requests by using the accept program.

`lp: destination "dest" non-existent`

The named destination does not exist. To display a list of the valid destinations, enter "lpstat -a".

`lp: request not accepted`

Errors have been found when collecting the list of files to print in the request. Error messages giving the FMP errors encountered should already have appeared. The request was not created.

`lp: couldn't allocate a sequence number:`
      fmp error message

An FMP error was encountered when allocating a sequence number for the request ID. The second line indicates the FMP error. The request was not created.

`lp: Invalid # of copies:` *parm*

An invalid number of copies was specified in the −n option. It must be an integer from 1 to 32767.

`lp: invalid priority:` *parm*

An invalid priority was specified in the −p option. It must be an integer from 0 to 7.

`lp: missing trailing quote:` *parm*

A matching double quote (") was not given for the parameter in the −t option.

`lp: unknown option:` *parm*

An unknown option name was given.

`.p: warning: lpsched is not running`

The request was created, but the scheduler (lpsched) is not running. The request will not be printed until the scheduler is started.

`lp: warning:` *violation* `error scheduling lpsched`

The lpsched program could not be scheduled.

`lp: warning:` *fmp error* `/usr/spool/lp/hostname`

An FMP error was encountered when retrieving the local host name from the named file.

lpadmin

Purpose:	Used to add or modify a printer, delete a destination, or set the system default destination.

Syntax:

lpadmin -p*printer setopts*	add or modify a printer
lpadmin -x*dest*	delete a destination
lpadmin -d [*dest*]	set system default destination

Only one of the −p, −x, or −d options must be given.

-p*printer*   Sets printer name to which *setopts* options apply. If the printer does not already exist, it is added to the spool system. The printer name may be up to 14 characters long, may not contain a dash (−), and must be a legal file name. *setopts* are defined below.

-x*dest*   Deletes *dest* from spool system; may name either a class or printer. If *dest* names a printer and that printer is the only member of a class, that class is removed also.

-d [*dest*]   Sets system default destination; may be the name of a printer or a class of printers. If no *dest* is supplied, no system default destination will be in effect.

Options (*setopts*)

The following options apply to the printer named in the −p option:

-c*class*   Make the printer a member of the named class (up to 14 characters long); the class is created if it does not already exist. A printer may be a member of only one class. If printer is already a member of a class, it must be removed from that class, using the −r option, before a new class may be specified. Remote printers may not be made class members.

-e*printer*   Copy an existing printer's model file to be the new printer model file for the printer.

-g*priority*   Set default priority for the printer (0 to 7). If not given, the default priority for the printer is 0.

-i*file*   Copy the named file to be the new printer model file for the printer.

-m*model*   Copy the named printer model file to be the new printer model file for the printer. Must be an existing model file name within the /usr/spool/lp/model/ directory.

$-r$*class*	Remove the printer from the named class. If the printer was the last member of that class, the class is removed.
$-v$*lu*	Set LU number of printer device (0 to 255). Ignored for remote printers. If not specified, LU is 0.

The following *setopts* are used for remote printers and network peripherals.

$-$orm*host*	Identify the name of the remote machine where a remote printer resides. For network peripherals, such as those on JetDirect cards, this is the name of the network peripheral. In general, this is the name in the /etc/hosts file that may be mapped to the IP address of the remote host or network peripheral.
$-$orp*printer*	Identify the name of a remote printer as known on the remote machine; may be a class name on the remote host. The name will be converted to all lowercase characters, and may be up to 14 characters long. If not specified, the local printer name will also be used as the remote name. For network peripherals, no value for this option is required because this name is not used in this case. However, entering a meaningful name that adds to the information shown by the lpstat $-v$ option report is suggested.

Only one of the $-e$, $-i$, or $-m$ options may be specified to select the printer model file. If a new printer is added without specifying one of these options, the default for local printers generic; the default for remote printers is rhpux.

You must be a superuser to run lpadmin.

Returns:

$RETURN1 = the number of errors encountered; zero if none

Error Messages

`lpadmin: Expected "-p", "-x", or "-d":` *parm*

The first option in the runstring must be one of these options.

`lpadmin: printer names may not contain dash (-)`

A dash (−) is illegal in printer names.

`lpadmin: invalid priority:` *parm*

An invalid priority was given in the −g option. Priorities must be an integer from 0 to 7.

`lpadmin: printer` *"printer"* `is not a member of class` *"class"*

The −r option was given to remove the printer from a class, but the printer is not a member of that class.

`lpadmin: printer` *"printer"* `is already a member of class` *"class"*

The −c option was given to insert the printer into a class, but the printer is already a member of a class. A printer may be a member of only one class.

`lpadmin: default printer model is` *"model"*

No −m option was specified, so `lpadmin` chose a default printer model for you.

`lpadmin: more than one model interface type chosen (-e/-i/-m)`

Only one of the −e, −i, or −m options may be given.

lpalt

Purpose: Alters parameters of an existing request, such as request priority, number of copies, and so forth.

Syntax: lpalt *id* [-*option*] [-*option*] ...

id   Request ID of the request to be altered. The request must be present on the local system; remote requests cannot be altered. You must be a superuser to alter a request that you do not own.

option   One or more of the options described below.

Options

-d*dest*   Change destination of the request, either a printer name or the name of a class. If a class is given, the files are printed on the first available printer in that class. When this option is used, a new request ID is assigned to the request. The new request ID is printed to the terminal and returned in $RETURN_S. The new request ID is a local request ID, even if the altered request originated on a remote host.

-m   Send mail to the user that created the request when the request is printed.

-n*copies* Change the number of copies to print of each file; must be an integer from 1 to 32767.

-o*option* Change the interface-specific options. More than one −o option may be given to pass more than one option. Existing options on the request are removed. See the documentation on the destination printer interface program or routine for the options that it supports.

          Because RTE case folds the lp runstring by default, and because most options are lowercase, lpalt reverses the upper or lowercase of each character in the option. For example, if a printer interface accepts option "BSDp", the lpalt runstring from CI might be:

```
CI> lpalt mach-5 -o`bsd`p
```

-p*priority* Change priority of the request, integer from 0 to 7.

-s	Suppress informational messages.
-t"*title*"	Change title to be printed on the banner page; must be less than 80 characters and must be surrounded by double quotes if it contains blanks or commas. When executing lpalt from CI, use backward quotes to prevent upshifting and comma-insertion of the title.
-w	When the request is printed, write a message on the terminal of the user that created the request.

Returns:

$RETURN1 = number of errors encountered; zero if none
$RETURN_S = new request ID if destination is changed

Error Messages

lpalt: can't accept requests for destination *"dest"* -
    reason for rejection

The destination rejects new submissions. The second line gives the reason set by the system manager when the destination was made to reject submissions. The destination can be made to accept requests by using the accept program.

lpalt: destination *"dest"* non-existent

The named destination does not exist. To display a list of the valid destinations, enter "lpstat -a".

lpalt: couldn't allocate a sequence number:
    fmp error message

An FMP error was encountered when allocating a sequence number for the request ID. The second line gives the FMP error.

lpalt: you do not own *requestid*

You do not own the request being altered. You must be a superuser to alter another user's request.

lpalt: request *requestid* is active on *printer*

The request is actively printing on a printer. Active requests may not be altered. To alter the request, you may disable the printer on which it is printing, alter the request, and re-enable the printer. This will reprint the request in its entirety.

`lpalt:` `no` `such` `request` *"requestid"*

The given request ID does not exist. Enter "`lpstat -o`" for a list of requests.

`lpalt:` `invalid` `#` `of` `copies:` *parm*

An invalid number of copies was specified in the −n option. It must be an integer from 1 to 32767.

`lpalt:` `invalid` `priority:` *parm*

An invalid priority was specified in the −p option. It must be an integer from 0 to 7.

`lpalt:` `missing` `trailing` `quote:` *parm*

A matching double quote (") was not given for the parameter of the −t option.

`lpalt:` `unknown` `option:` *parm*

An unknown option name was given.

`lpalt:` `warning:` `lpsched` `is` `not` `running`

The request was modified, but the scheduler (`lpsched`) is not running. The request will not be printed until the scheduler is started.

`lpalt:` `warning:` *violation* `error` `scheduling` `lpsched`

The `lpsched` program could not be scheduled.

`lpalt:` `warning:` *fmp error* `/usr/spool/lp/hostname`

An FMP error was encountered when retrieving the local host name from the named file.

lpfence

Purpose: Sets the fence priority for a printer.

Syntax: lpfence *printer priority*

Priority values are from 0 (lowest priority) to 7 (highest priority).
Only requests with the given priority or greater are printed on the
named printer. A request that could otherwise be sent to the printer,
but that has a priority lower than the fence, is sent to the printer once
the priority fence is lowered (or the request may be sent to another
member of the destination class).

This command affects only the local operation of a remote printer.
The fence setting on the remote host may be different than that of the
local system.

You must be a superuser to run lpfence.

Returns:

   $RETURN1 = the number of errors encountered; zero if none.

lpmove

Purpose: Moves requests from one destination to another.

Syntax: lpmove *id* [*id* ...] *todest*
             or
             lpmove *fromdest* *todest*

id   Request ID, as reported by "lpstat -o".

fromdest   Names of destinations, either printer names or class
todest   names.

The first usage above moves one or more requests by request ID.

The second usage above moves all requests for a destination to a new destination. Former destination is then set to reject new submissions.

Requests may be moved to a destination that is rejecting new submissions; lpmove prints a warning in this case.

You must be a superuser to move requests that do not belong to you, or to move all requests for a destination.

Error Messages

lpmove: warning: destination *"dest"* is not accepting
requests

> The destination to which requests are being moved rejects new submissions. The requests are moved to the destination anyway.

lpmove: destinations are identical

> The "from" and "to" destination names are the same.

lpmove: you do not own *"requestid"*

> The named request belongs to another user. You must be a superuser to move another user's request.

lpmove: request *"requestid"* is already destined for
dest

> The named request is already destined for the new destination.

lpmove: *"parm"* is not a request id

> A parameter was expected to be a request ID but that request was not found.

destination *dest* is not accepting requests

> This message informs you that the destination from which all requests have been moved now rejects new submissions. This message does not indicate an error condition.

lpsched

Purpose: Allows LP spool system to print requests (central printer scheduler for LP spool). It is normally executed in the welcome file at boot time.

Syntax: `lpsched [pq]`

`pq`   Optional parameter that causes a running `lpsched` to process the remote job queue immediately.

You must be a superuser to run `lpsched`.

`lpsched` takes requests from the LP spool system queue and sends them to the appropriate printer. When `lpsched` is not running, new requests may be created, but the requests are not printed locally or sent to remote printers until `lpsched` is started. `lpsched` is shut down by `lpshut`.

`lpsched` need not be scheduled without wait via the CI XQ command; it allows the scheduling program to continue after `lpsched` initializes.

`lpsched` keeps a log of its activity in file `/usr/spool/lp/log`.

If file `/usr/spool/lp/jobdone.cmd` exists, `lpsched` runs CI with wait on it each time a request successfully finishes printing, before the request files are purged. The request ID of the request printed is passed in $1.

Returns:

$RETURN1 = number of errors encountered; zero if none.

Error messages:

`lpsched: scheduler was already active`

An attempt was made to run `lpsched` when `lpsched` was already started.

`lpsched: invalid schedule`

`lpsched` was run with invalid parameters.

`scheduler is running`

This message informs you that `lpsched` was successfully started. It does not indicate an error condition.

lpshut

Purpose: Shuts down the lpsched program.

Syntax: lpshut

All printing activity at the time of shutdown is halted. No further printing occurs until the scheduler is restarted by running lpsched. Each print request that is actively printing at the time of shutdown is reprinted in its entirety when lpsched is restarted.

You must be a superuser to run this program.

Returns:

$RETURN1 = the number of errors encountered; zero if none.

Error Messages

lpshut: scheduler not running

The lpsched scheduler was already shut down.

scheduler stopped

This message indicates that lpsched was successfully shut down. It does not indicate an error condition.

lpstat

Purpose: Prints status information about the LP spool system.

Syntax: `lpstat` [*-option* | *request_id*] ... [> [>]*file*]

Each parameter not starting with a dash (−) or greater-than (>) must be a request ID. The status of that request is shown, in the same format as if "−*orequest_id*" were given.

>*file* redirects output to that file, overwriting an existing file.

>>*file* appends output to an existing file.

If neither of these are given, output is printed to scheduling terminal.

Many of the options listed below take an optional parameter *list*. If given, this parameter may be either:

- single word not containing blanks or commas (e.g., `-plj3`)

- list of words separated by blanks or commas and enclosed in double quotes (*""*) (e.g., `-p"lj3, magiclj"`)

Options

-a [*list*]   Display the acceptance status of destinations for requests. *list* is a list of printer names and/or class names. By default, all destinations are shown.

Note that for remote printers, only the status of the printers on the local system is shown. The acceptance status of printers on remote hosts is unknown. A remote printer that accepts requests will accept submissions on the local host for transfer to the remote host. However, the acceptance status of the printer on the remote host may be to reject submissions. In this case, the local host will place the remote printer in retry status.

-c [*list*]   Display class names and printers in those classes. *list* is a list of class names. By default, all classes are shown.

-d   Display system default destination. If a request is entered using the `lp` program and no destination is specified, then by default, the request is sent to this destination.

-i   Inhibit display of remote status; only local status is shown. If not given, `lpstat` queries remote hosts for status of requests on remote printers when −o or −u option given.

-o [*list*] Display status of output requests. *list* is a list of
destination names and/or request IDs. To select the
information shown by user name, see the −u option. By
default, all requests are shown.

If *list* names a remote printer, both local and remote
status may be displayed. Remote status display may be
inhibited by specifying the −i option.

-p [*list*] Display status of printers. *list* is a list of printer names.
By default, all printers are shown.

For remote printers, only the status of the printers on the
local system is shown. The full status of printers on
remote hosts is unknown. A remote printer that is
enabled on the local system will transfer requests to the
remote host. However, the printer on the remote host
may be disabled.

-r Display whether or not lpsched is running.

-s Display status summary; same as "−r −d −c −v".

-t Display all status; same as "−s −a −p −o".

-u [*list*] Display status of output requests by user name. *list* is a
list of user login names, optionally qualified by an at sign
(@) and a host name. By default, all requests are shown.
To select the information shown by other criteria, see the
−o option. The report shown is identical to that shown
by the −o option.

If remote printers exist on the system, both local and
remote status may be displayed. Remote status display
may be inhibited by specifying the −i option.

-v [*list*] Display the following device information for printers:

- If the printer is local, show the device LU number.

- If the printer is remote, show the remote host name
and the name of the printer on that host.

- If the printer is a network peripheral, as on an HP
JetDirect card, network peripheral name is shown.

list is a list of printer names. By default, all printers are
shown.

Error Messages

`lpstat: missing trailing quote: `*`parm`*

A matching double quote was not supplied for a *list* parameter that began with a double quote.

`lpstat: unknown option `*`"parm"`*

An unrecognized runstring option was given.

`lpstat: no printers`

No printers exist in the system. Printers may be added using `lpadmin`.

`lpstat: printer `*`"printer"`*` non-existent`

The named printer does not exist. Use "`lpstat -p`" to display the names of the valid printers.

`lpstat: no classes`

No class names exist on the system. Classes may be added using `lpadmin`.

`lpstat: class `*`"class"`*` non-existent`

The named class does not exist. Use "`lpstat -c`" to display the names of valid classes.

`lpstat: no destinations`

No destinations exist in the system. Destinations may be added using `lpadmin`.

`lpstat: destination `*`"dest"`*` non-existent`

The named destination does not exist. Use "`lpstat -a`" to display the names of valid destinations.

`lpstat: `*`"parm"`*` is not a request id`

A parameter was given, not as an option argument, that is not a valid request ID.

reject

Purpose: Prevents print requests from being submitted to the named destinations.

Syntax: reject [-r["*reason*"]] *dest* [*dest* ...]

-r["*reason*"]

Specifies a reason for preventing submissions. Reported by lp when an attempt is made to enter a request for the destination, and by lpstat. *reason* must be less than 80 characters, and should be surrounded by double quotes. From CI, use backward quotes to prevent upshifting and comma-insertion. *reason* will apply to all destinations named after the −r option; more than one −r may be specified, intermixed with destination names, to associate different reasons with different destinations. If not specified, the reason string "reason unknown" is used.

No print jobs may be entered for a destination rejecting submissions until the accept program is run for that destination. New printers and classes added to the LP spooler initially reject submissions.

You must be a superuser to run this program.

The accept and reject programs affect only the local spool system. Only the local operation of a remote printer is modified. A remote printer that rejects submissions will not accept requests for transfer to remote host; operation of the printer on remote host is unaffected.

Returns:

$RETURN1 = the number of errors encountered; zero if none.

Error Messages

reject: missing trailing quote: *parm*

A leading quote was found for the "reason" string, but no matching trailing quote was found.

reject: destination *"dest"* was already not accepting requests

The named destination already rejected new submissions.

reject: destination *"dest"* non-existent

An invalid destination was specified.

8

System Library Subroutines

Parameters

The following parameters are used throughout this section. They are not described under the routines that use them unless additional information is required for that routine.

prog	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.)
idcb	Data Control Block (DCB), an array of $16 + (n * 128)$ words. For file types 0 and 1, n is a nonnegative integer; for other file types, n is positive.
ierr	Error return, a 1-word variable in which a negative error code is returned, or zero if successful.
name	3-word integer array containing a 5-character segment name:

name(1)	1st two characters
name(2)	2nd two characters
name(3)	last character in upper 8 bits (lower byte is insignificant).

ABREG (Return A- and B-Registers)

Purpose:	Return contents of A-Register and B-Register.
Syntax:	CALL ABREG (*areg*, *breg*)
areg	1-word variable that returns contents of A-Register.
breg	1-word variable that returns contents of B-Register.

CHNGPR (Change Program Priority)

Purpose: Set the priority of calling program.

Syntax: *ierr* = CHNGPR (*prio*)

  ierr   Integer that returns the error. If nonzero, an error occurred.

  prio   Integer that specifies the new priority of the program. Can be set to a value between 1 and 32767, inclusive.

CLRQ (Class Management)

Purpose: Assign ownership to a class number.

Syntax: CALL CLRQ (*func*, *class* [, *pram1*])

  func   Class management control function word. Value is 1, 2, or 3, with optional bits set:

     Bit 15 = no-wait bit
     Bit 14 = no-abort bit
     Bit 12 = save data bit; valid only when the function is 3.

     1   Class ownership is assigned

     2   Flushes class requests and deallocates the class specified in *class* parameter.

     3   Flushes class requests on LU designated by *pram1*.

  class   Class number.

  pram1   Call-dependent parameter used to describe program name or LU.

Returns:

   Bit 15 set:

     A-Register   Set to -1 if no class numbers available, set to 0 if request completed without error.

   Bit 14 set:

     A-Register   Contains first two ASCII characters of the 4-character error message.

     B-Register   Contains second two ASCII characters of the message.

CNUMD and CNUMO (Convert Number to String)

Purpose: Convert a positive integer binary number to ASCII.

Syntax: CALL CNUMD (*numb*, *bufr*) (for decimal)

                 CALL CNUMO (*numb*, *bufr*) (for octal)

numb   Unsigned 16-bit integer to be converted to ASCII (in base 10 for CNUMD; base 8 for CNUMO).

bufr   3-word integer array where ASCII representation of *numb* will be stored.

CPUID (Get CPU Identification)

Purpose: Returns the CPU type.

Syntax: *icpu* = CPUID( )

icpu   Set to one of the following values:

Value	CPU Type
2	A600
3	A700
4	A900
5	A600+
10	A990

CPUT (Put Character in Buffer)

Purpose: Put specified character into destination buffer set up by SETDB.

Syntax: CALL CPUT (*char*)

char   Character to be put into left byte of destination buffer (FORTRAN 1H format).

EQLU (Get Interrupt LU)

Purpose: Return LU of interrupting device that scheduled
               program.

Syntax: *lu* = EQLU (*zlu*)

  lu   Returns number of first LU assigned to the DVT, or 0 if
               not found.

  zlu   Same as *lu*.

FCONT (Nondisk I/O Control Functions)

Purpose: Control I/O functions on nondisk type 0 file (same
               functions as ESC 3).

Syntax: CALL FCONT (*idcb*, *ierr*, *icnwd*, *pram**4)

  icnwd   See EXEC 3 call.

  pram*4   See EXEC 3 call.

Returns:

    A-Register   EXEC error code.

    B-Register   Device status.

FTIME (ASCII Time Request)

Purpose: Return a string in the form:

               "1:27 PM THU., 21 FEB., 1985"

Syntax: CALL FTIME (*bufr*)

  bufr   15-word integer buffer to which FTIME returns ASCII
               time string.

GETST (Get Runstring)

Purpose: Remove parameter string from a program's command
               string storage area.

Syntax: CALL GETST (*bufr*, *bufln*, *tlog*)

   bufr   User-defined buffer large enough for parameter string.

   bufln   Length of *bufr*, either *n* word or −2*n* characters
               (recommended length = 40 words).

   tlog   Transmission log; number of words or characters
               actually present in command string.

IDCLR (Clear ID Segment)

Purpose: Remove calling program from system by clearing (and
               deallocating) its ID segment.

Syntax: CALL IDCLR ( )

IDGET (Get ID Segment Address)

Purpose: Return ID segment address of program name.

Syntax: *iaddr* = IDGET (*name* [, *ses*])

   iaddr   User-supplied variable in which the ID segment
               address of the program name is returned; 0 if the
               program does not have an ID segment in memory.

   name   3-word integer buffer containing 5-character name.

   ses   Optional session number. Default = caller's session
               number.

IDINFO (Return ID Segment Information)

Purpose: Retrieves information from the ID segment of the
               calling program.

Syntax: *ierr* = IDINFO (*idaddr*, *pname*, *astat*, *mefstat*, *piaddr*)

ierr   Integer that returns an error. If negative, the ID
               segment address is invalid or the ID segment currently
               does not contain a program.

idaddr   Integer that specifies the address of the ID segment.

pname   3-word integer array that is set to the name of the
               program. The array is left-justified blank filled.

astat   Integer that returns the state of the program if running
               on an A-Series computer. Parameter must be set to 0
               prior to the call to IDINFO.

mefstat   Integer that returns the state of the program if running
               on an M/E/F-Series computer. Parameter must be set
               to 0 prior to the call to IDINFO.

piaddr   Integer that returns the address of the parent
               program's ID segment. If negative, the parent is
               waiting; if positive, the parent is not waiting.
               Parameter must be set to 0 prior to the call to IDINFO.

IFBRK (Was Break Entered?)

Purpose: Test break flag and clear it if set.

Syntax: *ibk* = IFBRK ( )

ibk   1-word variable containing a returned value of 0 (not
               set) or −1 (set).

IFTTY (LU Interactive?)

Purpose: Ascertain whether or not an LU is interactive.

Syntax: *int* = IFTTY (*lu*)

int   −1 = LU is interactive.
               0 = LU is not interactive.

INMAR (Build NAMR)

Purpose: Inverse of the NAMR routine, building a character string in the NAMR format from an input array (parameter buffer).

Syntax: CALL INAMR (*ipbuf*, *ubuf*, *maxb*, *istart*)

ipbuf Parameter buffer; 10-word array in which the *namr* parameters are stored; same format as for *ipbuf* in the NAMR routine.

ubuf Returned user buffer; output array that will contain the character string generated by INAMR.

maxb *ubuf* length in characters; 1-word variable.

istart Starting character position; 1-word variable; the position within the user buffer to start the namr string.

INPRS (Inverse Parse)

Purpose: Convert parsed buffer of data back into its original ASCII form.

Syntax: CALL INPRS (*rbufr*, *pnum*)

rbufr Buffer containing parsed string.

pnum Number of parameters in parsed string.

KCVT (Convert Number to 2 Characters)

Purpose: Convert positive integer to base and return last two equivalent ASCII digits.

Syntax: digits = KCVT (*num*)

digits 2-character buffer.

num 1-word integer in range 0 to 65535 decimal.

KHAR (Character Routines)

Purpose: Get next character from source buffer set up by SETSB.

Syntax: *char* = KHAR (*dchar*)

  char   Integer variable where character is returned (0 if no more characters in source buffer).

  dchar   Same as *char*.

LIMEM (Find Available Memory Unit)

Purpose: Allow program to use and manage the memory between the end of its code and the end of the memory available to it.

Syntax :   CDS or non-CDS programs without overlays:

            CALL LIMEM (*code*, *fwam*, *words*)

            Non-CDS programs with overlays:

            CALL LIMEM (*code*, *fwam*, *words*, *curnt* [, *cwrds*])

  code   Ignored.

  fwam   First word of available memory after the largest program overlay.

  words   Number of words available after *fwam*.

  curnt   First word available after current overlay.

  cwrds   Number of words available after *curnt*.

LOGIT (Send Logging Message)

Purpose: Log an error message to terminal log file.

Syntax: *ierr* = LOGIT (*string*, *len*)

  ierr   Integer that returns the error code. If negative, an error occurred. (Note that a positive value does not guarantee that the message was logged.)

  string   Integer buffer that contains the message (128 words maximum).

  len   Integer that contains the length of the message in words.

LOGLU (Get Log Device LU)

Purpose:	Return LU number suitable for EXEC calls to terminal.
Syntax:	*lu* = LOGLU (*reallu*)
lu	Always 1.
reallu	Real LU number of scheduling terminal (255 maximum).

LURQ (Lock LU)

Purpose:	Allow program to exclusively dominate (lock) an I/O device.
Syntax:	CALL LURQ (*option* [, *luary* [, *numlus* [, *keynum*]]])
option	Control parameter, in octal:

    0xy000B   Unlocks specified LUs.

    1xy000B   Unlocks all LUs currently locked by the program.

    0xy001B   Locks specified LUs with wait.

    1xy001B   Locks specified LUs without wait.

    x = 4 to set bit 14, else x = 0 (no-abort option).

    y = 2 to set bit 10, else y = 0 (override spool node lock).

luary	Integer array of LUs to be locked or unlocked.
numlus	Number of LUs in *luary*.
keynum	Returns key number, which allows multiple programs to share one lock.
Returns:	A-Register:

0   = Lock successful.
1   = one or more LUs are already locked.

MESSS (Issue OS Command From Program)

Purpose: Execute interactive commands available from "RTE:" (or "SYSTEM>") prompt.

Syntax: *ic* = MESSS (*bufr*, *count* [, *lu*])

ic   Returns negative character count of message returned in *bufr*, or 0 if no message is returned.

bufr   User-defined buffer, at least 72 characters. Contains command string before call; overlaid by return error or status message.

count   Number of characters in *bufr*.

lu   Meaningful for RU or XQ requests only. Returns LOGLU value for scheduled program.

NAMR (Parse Name)

Purpose: Parse an array (buffer) of any length and return up to seven subparameters.

Syntax: CALL NAMR (*ipbuf*, *ubuf*, *maxb*, *istart*)

ipbuf   Parameter buffer; 10-word array in which up to seven subparameters are returned from the input string.

First subparameter = Word 1 through 3:

Word 1:   If type = 0, word 1 has a value of 0.

If type = 1, word 1 is a 16-bit twos complement number.

If type = 3, word 1 contains characters 1 and 2.

Word 2:   If type = 0 or 1, word 2 has a value of 0.

If type = 3, word 2 contains characters 3 and 4 or trailing blanks.

Word 3:   If type = 0 or 1, word 3 has a value of 0.

If type = 3, word 3 contains characters 5 and 6 or trailing blanks.

Word 4:   Gives the parameter types of all seven sub-parameters, in two-bit pairs:

0 = null
1 = integer numeric
2 = (no meaning assigned)
3 = ASCII

Bits 0 and 1   = type of first sub-parameter.
Bits 2 and 3   = type of second sub-parameter.
Bits 4 and 5   = type of third sub-parameter.
Bits 6 and 7   = type of fourth sub-parameter.
Bits 8 and 9   = type of fifth sub-parameter.
Bits 10 and 11   = type of sixth sub-parameter.
Bits 12 and 13   = type of seventh sub-parameter.

Words 5 through 10 have the format of word 1. They are (items in parentheses are the corresponding file namr subparameters):

Word 5   = 2nd subparameter (security code).
Word 6   = 3rd subparameter (cartridge reference).
Word 7   = 4th subparameter (file type).
Word 8   = 5th subparameter (file size).
Word 9   = 6th subparameter (record size).
Word 10   = 7th subparameter.

ubuf   User buffer input array to be parsed.

maxb   *ubuf* length in characters; 1-word variable.

istart   Starting character position; 1-word variable; will be updated with each call to NAMR.

.OPSY (Return Unique System Value)

Purpose: Return value unique to operating system on which
             program is running. Note that the RTE-A .OPSY value
             may change in future versions of RTE-A.

Syntax: JSB .OPSY

Returns: A-Register

             -7 = RTE-MI
             -15 = RTE-MII
             -5 = RTE-MIII
             -9 = RTE-IV
             -29 = RTE-XL
             -31 = RTE-L
             -37 = RTE-A (Pre-Rev. 2440)
             -45 = RTE-A.1
             -53 = RTE-A (Rev. 2440 through 4010)
             -61 = RTE-A (Rev. 5000 through 5270)
             -125 = RTE-A (Rev. 6000 and later)

OVF (Report and Clear Overflow Bit)

Purpose: Return value of overflow bit; then clear the bit.

Syntax: *iovr* = OVF ( )

iovr -1 = Overflow bit set.
             0 = Overflow bit clear.

PARSE (Parse ASCII String)

Purpose: Parse an ASCII string of up to eight parameters.

Syntax: CALL PARSE (*bufr*, *crcnt*, *rbufr*)

bufr   Buffer containing string to be parsed.

crcnt   Number of characters in *bufr* (supplied by user).

rbufr   Parsed string, 33 words long. A set of 4 words describes
             each parameter parsed:

Word	Entry		
1	FLAG WORD	0 = NULL	
		1 = NUMERIC	
		2 = ASCII	
2	VALUE(1)	0 if null, parameter value if numeric, first 2 characters if ASCII.	
3	VALUE(2)	0 if null or numeric, otherwise 3rd and 4th characters.	
4	VALUE(3)	0 if null or numeric, otherwise 5th and 6th characters.	
33		Number of parameters passed.	

PNAME (Get True Program Name)

Purpose: Copy program name from caller's ID segment to a buffer in the program.

Syntax: CALL PNAME (*bufr*)

bufr   6-character buffer.

PRTN and PRTM (Pass Parameters)

Purpose: Pass parameters to the program that scheduled it with wait. (PRTM provides compatibility with RTE-6/VM.)

Syntax: CALL PRTN (*prams*)

CALL PRTM (*prams*)

prams   5-element array, one word per element. 5 parameters can be passed (or 4 with PRTM, whose first parameter is meaningless).

REIO (Reentrant I/O)

Purpose:	Allow program to be swapped (with buffered I/O) while waiting for input.
Syntax:	CALL REIO (*ecode*, *cntwd*, *bufr*, *bufln* [, *p3* [, *p4* [, 0 , 0 , *keynum*]]])
ecode	Request code: 1 = Read 2 = Write 3 = For control
cntwd	Control word, exactly as in EXEC 1, 2, and 3.
bufr	Integer array. Data is moved to or from the array depending on the value of *ecode*.
bufln	Integer that specifies the length of the buffer in words (positive value) or in bytes (negative value).
p3, *p4*	Integers that specify driver-dependent parameters.
keynum	Key number of locked LU.

Returns:

Normal I/O

A-Register	Word 6 of DVT.
B-Register	Positive number of words or characters actually transmitted.

Output requests to buffer device:

A-Register	Meaningless.
B-Register	Meaningless.

RMPAR (Recover Parameters)

Purpose:	Recover parameters passed to calling program.
Syntax:	CALL RMPAR (*prams*)
prams	5-word array. Returns up to 5 parameters.

RNRQ (Allocate Resource Number)

Purpose: Allocate and manage resources.

Syntax: CALL RNRQ (*cntwd* , *rn* , *stat*)

cntwd   Control word, which defines resource number use.
            Format:

            Bit 15   No wait
                14   No abort
                 5   Deallocate RN
                 4   Allocate globally
                 3   Allocate locally
                 2   Unlock
                 1   Lock globally
                 0   Lock locally

rn   One-word integer that returns the allocated RN on
            allocation requests, or specifies the RN for lock,
            unlock, or deallocation requests.

stat   Returns status:
                −1 = Invalid request
                 0 = Normal deallocate return
                 1 = RN is allocated/unlocked as requested
                 2 = RN is locked locally to caller
                 3 = RN is locked globally
                 4 = No RN available now
                 6 . = RN is not allocated or RN locally locked . .
                        to another program
                 7 = RN was locked when request was made

SEGLD (Segment Load)

Purpose: Load segment of calling program from disk into an overlay area in memory provided by the program, and transfer control to the segment's entry point.

Syntax: CALL SEGLD(*name*,*ierr*[,*p1*[,*p2*[,*p3*[,*p4*[,*p5*]]]]])

  name   3-word array containing 5-character overlay name.

  ierr   −6 if overlay cannot be loaded or calling program is CDS program.

  p1-p5   Optional user-defined integer parameters that are passed to the overlay.

SEGRT (Segment Return)

Purpose: Return to the instruction following SEGLD call in the main program.

Syntax: CALL SEGRT( )

SETDB (Set Up CPUT/ZPUT Destination Buffer)

Purpose: Set up destination buffer for CPUT and ZPUT.

Syntax: CALL SETDB(*dbufr*,*chcnt*)

  dbufr   Integer destination buffer.

  chcnt   Number of characters in *dbufr*.

SETSB (Set Up CPUT/ZPUT Source Buffer)

Purpose: Set up source buffer for KHAR.

Syntax: CALL SETSB(*bufr*, *chpos*, *bufln*)

bufr   Integer buffer containing string to be examined.

chpos   Current character position.

bufln   Number of characters in *bufr*.

SETTM (Set System Time)

Purpose: Set system time.

Syntax: *ierr* = SETTM(*hr*, *min*, *sec*, *mon*, *day*, *yr*)

ierr   0 = No error; time is set.
                −1 = Illegal parameter value.

hr   Hour, in 24-hour format.

min   Minute.

sec   Second.

mon   Month.

day   Day.

yr   Year, in four digits; at least 1976.

All parameters are integers.

SYCON (Writes Message to System Console)

Purpose: Write a message to the system console.

Syntax: CALL SYCON(*ibuf*, *ilen*)

ibuf   Buffer that contains the message to be written.

ilen   Length of *ibuf*; positive value indicates the number of words and a negative value indicates the number of characters.

XLUEX (Extended LU EXEC Call)

Purpose: Extended EXEC call. Accommodates up to 255 LUs.

Syntax: CALL XLUEX (*ecode , cntwd , bufr , buflen , prams*)

Parameters are same as for EXEC, except *cntwd*, which has the format:

Word 1:

Bit 15	OV bit
Bit 14	OS bit
Bit 13	0
Bit 12	WT bit
Bits 11-8	Reserved
Bits 7-0	LU

Word 2:

| Bits 15-6 | Same as *cntwd* in EXEC call. |
| Bits 5-0 | Reserved |

XQPRG (Load and Execute Program)

Purpose: Load and execute a program.

Syntax: CALL XQPRG (*idcb , icode , name , ifive , ibuf , il , iprtn ,
                    ierr , isecu , icr*)

icode   EXEC request code; 1-word variable used to schedule the program.

> 9 = Immediate schedule with wait
> 10 = Immediate schedule without wait.
> 23 = Queue schedule with wait.
> 24 = Queue schedule without wait.

ifive   Scheduling parameters; optional 5-word array to be passed to program.

ibuf   User buffer; optional array to be passed to program.

il   User buffer length; optional 1-word variable (+ words, − characters).

iprtn   Return parameters; optional 5-word array passed back from program.

ierr   0 = Successful execution, no scheduling errors.

    1 = Duplicate program name (*iprtn*(1) = 0).

2 = No ID segments available ($iprtn(1) = 14$).

3 = Program not found ($iprtn(1) = -6$ or -32).

4 = File open error other than "program not found" ($iprtn(1) = $ FMP error number).

5 = File close error ($iprtn(1) = $ FMP error number).

6 = RP error other than "duplicate program name"; no ID segments available ($iprtn(1) = $ FMP error number).

7 = Program busy ($iprtn(1) = 0$).

8 = Program was scheduled, but then aborted ($iprtn(1) = 100000B$).

10 = Illegal icode parameter (should be 9, 10, 23, or 24).

11 = Not enough parameters.

isecu   Security code; optional 1-word variable; must be specified to execute program in file created with a negative security code.

icr   CRN; optional 1-word variable; if zero, the first file found with the specified name will be executed; if set, only the file on the specified cartridge will be executed.

XQTIM (Time Schedule Program)

Purpose: Time schedule program.

Syntax: CALL XQTIM(*idcb*, *ierr*, *name*, *isecu*, *icr*, *ires*, *mult*, *itime*)

isecu   Security code; optional 1-word variable; must be
               specified to execute a program contained in a file
               created with a negative security code; may be omitted if
               the file was not protected at creation time.

icr   CRN; optional 1-word variable; if set, FMP searches
               that cartridge for the file; if omitted, it searches
               cartridge in the cartridge list order and executes the
               first file found with the specified name.

ires   Resolution code; optional 1-word variable, if omitted,
               zero is used.

mult   Resolution multiple; optional 1-word variable; program
               will be scheduled every *ires* * *mult* time intervals; if
               omitted, zero is used.

itime   Time parameters; optional 4-word array; specifies
               either absolute or relative starting time for program
               execution; for absolute starting time.

               Word 4   hours
               Word 3   minutes
               Word 2   seconds
               Word 1   tens of milliseconds
               Word 0   remove from time list

               For the relative time (in *ires* units) from the XQTIM
               call:

               Word 1 =   −Offset; this must be a negative number;
                          program will begin execution *ires* * offset
                          from now. Words 2 through 4 are
                          ignored.

XREIO (Extended LU REIO Call)

Purpose: Extended REIO call. Accommodates up to 255 LUs.

Syntax: CALL XREIO(*ecode*, *cntwd* [, *pram*4* [, 0 , 0 , *keynum*]])

Parameters are same as for REIO, except *cntwd*, which is the same as for XLUEX.

ZPUT (Put String in Buffer)

Purpose: Put string in destination buffer set up by SETDB.

Syntax: CALL ZPUT(*zbufr*, *frstc*, *noc*)

  zbufr   Integer buffer containing string to be stored in
                destination buffer.

  frstc   Position of first character to be put in destination
                buffer.

  noc   Number of characters to be put in destination buffer.

EXEC Calls

EXEC 1, 2 (Read/Write)

CALL EXEC (*ecode*, *cntwd*, *bufr*, *bufln* [, *pram3* [, *pram4* [, 0 , 0 , *keynum*]]])

ecode   Request code: 1 = Read
                                      2 = Write

cntwd   The control word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BB	NB	UE	Z	x	TR	X	EC	x	BI			LU			

                BB   Is the device drive bypass bit
                NB   Forces nonbuffered operation
                UE   Allows user device error handling
                Z   Specifies optional parameters *pram3* and
                                  pram4 as a buffer and buffer length
                TR   Establishes transparency mode in effect
                EC   Establishes echo of input
                BI   Selects ASCII or binary data transfer
                LU   Selects device by logical unit number
                X   Is defined for the appropriate driver

bufr   Buffer. For read operations (*ecode*=1), this is the array in which the system returns data. For write operations (*ecode*=2), *bufr* is the array containing the data to be written.

bufln   Buffer length. (+ words, − characters).

pram3   Optional parameter or buffer, if Z-bit is set.

pram4   Optional parameter or buffer length, if Z-bit is set.

keynum   The locked LUs key number, corresponding to the keynum parameter returned by LURQ.

Returns:

For reads and nonbuffered writes:

      A-Register   Word 6 of the DVT.
      B-Register   A positive number that is the number of words or characters actually transmitted.

Buffered writes:

      A-,B-Register Meaningless

EXEC 3 (I/O Device Status)

CALL EXEC(3,*cntwd*[,*pram1*[,*pram2*[,*pram3*[,*pram4*[,0,0,*keynum*]]]]])

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BB	NB	UE	Z	\multicolumn{6}{c}{Function}						LU				

BB	Driver bypass bit.
NB	Forces nonbuffered operation
UE	Allows user device error-handling
Z	Specifies optional parameters *pram3/4* as the Z-buffer and Z-buffer length, respectively.
Function	Specifies desired control function. See codes below.
LU	Selects device by logical unit number.

pram1, *pram2*	Driver-dependent parameters.
pram3, *pram4*	Driver-dependent parameters or the optional buffer and buffer length if the Z-bit is set.
0,0	Formal placeholders required if the next parameter is used.
keynum	Locked LUs key number.

Returns:

A	Device status in DVT word 6.
B	Device status in DVT word 17.

Function Codes: Octal bits 6-11.

The control function code (0-63B) as defined for each driver. The following is a partial list (refer to the *RTE-A Driver Reference Manual* for a complete list):

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 spaceing, use PRAM1 (CRT).
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 timeout, use PRAM1 (CRT).
23 Ignore further requests until:
   a) Device queue emppty.
   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 (CS/80) disks.
34 Block addressing (CS/80) disks.

EXEC 6 (Stop Program Execution)

CALL EXEC(6,*prog*[,*type*[,*pram*5]])

prog　　　　0 = Terminate the calling program.

To terminate a subordinate program, use the name of a 3-word character array containing the name (in uppercase) of the program to be terminated.

type　　　　Type of termination.

0　Normal completion (default).
1　Make the program dormant, save resources.
-1　Terminate serially reusable.
2　Terminate normally and remove program from time list.
3　Same as 2 and remove program's ID segment.

*pram*5　　　Up to 5 user-definable parameters to be passed to the calling program when it is next scheduled, retrieved by a call to RMPAR.

EXEC 7 (Program Suspend)

CALL EXEC(7)

EXEC 8 (Segment Load)

CALL EXEC(8,*name*[,*pram*5])

name	3-word integer array specifying the segment name in all uppercase.
*pram*5	Up to 5 user-defined parameters, retrieved by a call in the segment to RMPAR.

Returns:

On segment entry:

A-Register	Address of the (segment's) short ID segment.
B-Register	Address of the parameter list.

If the segment does not exist:

A-Register	ASCII characters "SC" (51503 octal).
B-Register	ASCII characters "05" (30065 octal).

EXEC 9, 10, 23, 24 (Program Scheduling)

CALL EXEC (*ecode*, *name* [, *pram**5 [, *bufr* [, *bufln*]]])

ecode	9 = Immediate schedule with wait. 10 = Immediate schedule without wait. 23 = Queue schedule with wait. 24 = Queue schedule without wait.
name	3-word integer array specifying the 5-character program name (in all uppercase).
pram*5	Up to 5 optional integer parameters.
bufr	Where data to be sent to or received from the child program is placed.
bufln	Length of *bufr* (+ words, − characters).

Returns:

A-Register

Returns from EXEC 9 or 10 requests:

> If the named program is dormant, it is scheduled and a zero is returned to the calling program in the A-Register. If the program is not dormant, it is not scheduled by these calls and a nonzero value is returned to the calling program in the A-Register.

Returns from EXEC 23 or 24 requests:

> Status will not be available in the A-Register (and the parent program will be impeded) until the scheduling request is honored.

> 0 if request succeeded.

B-Register   Meaningless.

EXEC 11 (Time Retrieval Request)

CALL EXEC(11, *time* [, *year*])

time   5-word integer array where the time value is returned:

time(1) = Tens of milliseconds
time(2) = Seconds
time(3) = Minutes
time(4) = Hours (0 to 23)
time(5) = Julian day of the year (such as 117)

year   year

EXEC 12 (Initial Offset Scheduling)

CALL EXEC(12, *name*, *units*, *often*, *delay*)

name   3-word integer array specifying the 5-character program name (in uppercase).

units   Resolution code specifying time units, and is used with often:

0 = remove from time list
1 = tens of milliseconds
2 = seconds
3 = minutes
4 = hours

often   Execution multiple, how often the program is to run.

delay   Initial offset, a negative number indicating the starting time of the first execution.

EXEC 12 (Scheduling Absolute Start Time)

CALL EXEC(12[,*name*[,*units*[,*often*[,*hour*[,*min*[,*sec*[,*msec*]]]]]]])

name	3-word integer array specifying the 5-character program name (in uppercase).
units	Resolution code; same as EXEC 12 (initial offset).
often	Execution multiple, how often the program is to run.
hour	Starting hour, 0 to 23 (default = 0).
min	Starting minute (default = 0).
sec	Starting second (default = 0).
msec	Starting tens of milliseconds (default = 0).

EXEC 13 (Device Status Request)

CALL EXEC(13,*cntwd*,*stat1*[,*stat2*[,*stat3*[,*stat4*]]])

Check status of I/O operations; returns operating system maintained information and driver supplied information.

cntwd   The control word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Z		0							LU					

stat1		Returned device status (DVT word 6).
stat2		Returned interface status (IFT word 6).
stat3,stat4	If Z=0	returned driver parameter words 1 and 2.
	If Z=1	buffer and length where information from the driver parameter area is returned.

EXEC 14 (String Passage Call)

CALL EXEC(14,*rcode*,*bufr*,*bufln* [,*prams*]))

rcode	Retrieve or write code: 1 = retrieve parameter string or buffer 2 = write buffer to parent 3 = write buffer to calling program
bufr	Integer buffer where the string is placed.
bufln	Buffer length, (+words, − characters); recommended length = 128 words.
prams	Integer array five words long that is copied to the program's temporary area for later retrieval by RMPAR call. This optional parameter is for use when *rcode* = 3.

Returns:

A-Register	Contains status information: 0 = operation successful 1 = no string found
B-Register	Positive number of words (or characters) transmitted.

EXEC 17, 18, 20 (Class Read, Write, Write/Read)

CALL EXEC (*ecode*, *cntwd*, *bufr*, *bufln*, *pram3*, *pram4*, *class* [, *uv* [, *keynum*]] ,

ecode 17 = Class Read.
18 = Class Write.
20 = Class Write/Read.

cntwd The control word:

15	14	13	12	11	10	9	8	7	6	5	0
BB	NB	UE	Z	x	TR	x	EC	x	BI	LU	

BB Is the device driver bypass bit
NB Forces nonbuffered operation
UE Allows user device error handling
Z   Indicates *pram3* and *pram4* are buffer and
    buffer length, respectively
TR Indicates transparency mode
EC Allows echo of input
BI Indicates ASCII or binary data transfers
LU Selects the device by LU number; an LU of
    zero indicates a program- to-program
    data transfer
X   Is defined for the appropriate driver

bufr Integer array used as a read/write buffer.

bufln Length of bufr (+ words, − characters).

pram3 Optional parameter or buffer address.

pram4 Optional parameter or buffer length.

class The class number, which can be set to zero to inform
the system that the program wants a class number
issued.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NW	SB	RT	CN												

NW No wait
SB Save class buffer
RT Rethread class bit
CN Class number

uv	User-defined variable maintained by the system and returned in the Class Get call. If the request is a class rethread, this variable represents the OCLAS parameter.
keynum	The key number for a locked LU.

Returns:

A-Register	0	Request successful (no error).
	−1	No class number currently available.
	−2	No memory now, buffer limit exceeded, or pending count is already 255.
B-Register	Meaningless.	

EXEC 19 (Class I/O Device Control)

```
CALL EXEC(19,cntwd,pram1,class[,pram2[,pram3[,pram4[,uv
                                          [,keynum]]]]])
```

cntwd   The control word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BB	NB	UE	Z	Function						LU					

BB	Bypasses the device driver.
NB	Forces nonbuffered operation.
UE	Allows user device error handling.
Z	Specifies optional parameters *pram3* and *pram4* as the buffer and buffer length, respectively.
Function	Specifies desired control function. (See EXEC 3).
LU	Selects device by LU number.

pram1	User-defined parameter.
class	Class number.
pram2	Driver-dependent parameter for certain control functions.
pram3 *pram4*	Driver-dependent parameters for optional buffer and buffer length if Z-bit is set.
uv	Optional user-defined parameter that can be retrieved in a future Class Get.
keynum	Optional parameter for locked LUs key number.

Returns:

A-Register	0	Request successful (no error).
	−1	No class number currently available.
	−2	No memory now, buffer limit exceeded, or pending count is already 255.

EXEC 21 (Class I/O Get)

CALL EXEC(21,*class*,*bufr*,*bufln*[,*rtn1*[,*rtn2*[,*rtn3*[,*uv*[,*to*]]]]])

class   The class number:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NW	SB	SC	CN												

NW	No wait
SB	Save class buffer
SC	Save class number bit
CN	Class number

bufr   User-defined integer array where information will be transferred.

bufln   Length of bufr (+ words, − characters).

rtn1   Corresponds to *pram3* from a read or write call.
            Corresponds to *pram1* from a control call.

rtn2   Corresponds to *pram4* from a read or write call.
            Corresponds to DVT word 17 after driver modification for a control call.

rtn3   Request code passed to the driver on an initial read, write, or control call.

1 if call was 17 or 20	(Read or Write/Read)
2 if call was 18	(Write)
3 if call was 19	(Control)

uv   User-defined variable returned from a previous read, write, or control call.

to   Class timeout or wait time.

Returns:

A-Register   If the A-Register = 10,000 then the Get has timed out.

             If data, bit 15 of the A-Register = 0, and the A-Register = device status word 6 of the Device Table (DVT)

             If no data, bit 15, 14 of the A-Register = 1,0 and the A-Register = the ones complement (−n-1) of the number of requests made to the class but not yet serviced by the driver (pending class requests).

B-Register   If data, the B-Register = transmission log (positive number of words or characters transmitted to the system buffer, depending on the original request).

             If no data, the B-Register = meaningless.

EXEC 22 (Program Swapping Control)

CALL EXEC(22,*option*)

option   Specifies alteration:

0   Data partition can be swapped.
1   Data partition cannot be swapped.
2   Code partition can be overlaid (CDS program only).
3   Code partition cannot be overlaid (CDS program only).

EXEC 26 (Memory Size Request)

CALL EXEC(26,*fword*,*nwords*,*lpart*[,*umap*[,*cmap*]])

fword   returns the address of the first available word behind the calling program.

nwords   returns the number of pages available between the last word of the program and the last word of the program address space.

lpart   returns the length in pages of the partition occupied by the program, including the user base page.

umap   For non-CDS programs, a 32-word array that returns a copy of the currently enabled user map. For CDS programs, contains current user data map.

cmap   For CDS programs, a 32-word array that returns a copy of the current user code map.

EXEC 29 (Retrieve ID Segment Address)

CALL EXEC(29,*name*,*session*,*idaddr*[,*search_flag*])

name   three-word integer buffer that contains the name of the
                program whose ID segment address is to be retrieved.
                The first unoccupied ID segment may be found by
                setting each element of *name* to zero.

session   number of the session with which *name* is associated.
                Setting *session* to 0 specifies the system session. This
                parameter is ignored if any of the following conditions
                are found to be true:

                • *name* is all zeroes,
                • *search_flag* is specified and has the sign bit set,
                • *name* specifies a program that is a system process or
                  is in the system session.

idaddr   set by EXEC 29 to the ID segment address of the
                program that was found. If a matching program is not
                found, *idaddr* is set to 0.

search_flag   if specified, and the sign bit of this parameter is set, it
                indicates that the *session* parameter does not have to
                match for an ID segment to match.

EXEC 37 (Signals)

The following are the EXEC 37 calls. The equivalent subroutine for each EXEC 37 call is also listed.

Set the Signal Handler Address

CALL EXEC(37,1,*handler*)
CALL SglHandler(*handler*)

    handler   is a pointer to the handler routine.

Set Signal Buffer Limits

CALL EXEC(37,2,*lo*,*hi*,*rtlo*,*rthi*)
CALL SglLimit(*lo*,*hi*,*rtlo*,*rthi*)

lo	Integer specifying the lower buffer limit for the program receiving signals.
hi	Integer specifying the upper buffer limit for the program receiving signals.
rtlo	Same as *lo* but applies to real-time programs (priority ≤ 40).
rthi	Same as *hi* but applies to real-time programs (priority ≤ 40).

Send a Signal to a Program

CALL EXEC(37,3,*name*,*ses*,*sig*,*buf*,*len*)

CALL SglKill(*name*,*ses*,*sig*,*buf*,*len*)

name	3-word integer buffer that specifies the name of the program that is to receive the signal.
ses	Integer identifying the session the program is in.
sig	Integer specifying the signal that is to be sent. If *sig* is zero, error checking is performed on the other parameters. The following are valid signal numbers.

Signal Number	Signal Type
4	Program Violation
14	Timer Completed
16	User Definable
17	User Definable
22	Class I/O Completion

buf	Integer buffer containing the signal-dependent buffer that should be passed to the program.
len	Integer identifying the length of the signal-dependent buffer in bytes.

Block Signals and Return Previous Set of Masked Signals

CALL EXEC(37,4,1,*mask*)

oldmask = SglSetMask(*mask*)

oldmask	Double integer that returns the previous set of masked signals.
mask	Double integer that contains the mask of the signals that are to be blocked.

Return Previous Set of Masked Signals and Block Signals in Addition to Current Signals

CALL EXEC(37,4,2,*mask*)

oldmask = SglBlock(*mask*)

oldmask	Double integer that returns the previous set of masked signals.
mask	Double integer that contains a mask of signals that are to be blocked from delivery.

Wait for a Signal to be Delivered to the Program

CALL EXEC(37,4,3,*mask*)
CALL SglPause(*mask*)

mask	Double integer specifying the signals that should be blocked from delivery while the program is paused.

Jump to Supplied Environment

CALL EXEC(37,5,*env*)
CALL SglLongJmp(*env*)

env	3-word integer array containing information set up by a SglSetJmp call or by the operating system calling a signal handler.

Set an Environment

CALL EXEC(37,6,*env*)
rtntype = SglSetJmp(*env*)

env	3-word integer array returning information about the present environment including the PC with a CDS mode indicator, the active code segment, and the current stack register (Q).
rtntype	Integer that returns 0 when SglSetJmp returns after setting an environment, or returns the contents of the A-Register at the time that environment is jumped to via SglLongJmp.

Return Integer Value Signifying Action to be Taken

(no equivalent EXEC 37 call)
action = SglAction(*sig*)

action	Integer specifying the action to take.
sig	Integer signifying the signal number.

EXEC 38 (Timer Signals)

Establish New Timer or Reset Existing Timer

CALL EXEC(38,0,*interval*)
error. = SetTimer(*interval*)

    interval   Double integer indicating the number of ticks between 0 and $2^{32}-1$ before the signal is generated. Each timer tick takes 10 ms. An interval of zero causes a timer signal immediately.

Returns zero if no error occurs, -1 if no interval is specified.

Return Number of Ticks Remaining Before a Timer Signal is to be Generated for the Calling Program

CALL EXEC(38,1,*ticks*)
error = QueryTimer(*ticks*)

    ticks   Double integer that, upon return, contains the number of ticks remaining for the calling program's interval. Each tick has a value of 10 ms.

Returns zero if no error occurs, -1 if no timer exists for the calling program.

Cancel Current Timer for the Calling Program

CALL EXEC(38,2,0)
error = KillTimer( )

Returns zero if the timer is canceled, -1 if no timer exists for the calling program.

EXEC 39 (Environment Variable Access)

Get the Value of an Environment Variable

CALL EXEC(39,1,*name*,*value*)

name   FORTRAN string descriptor that specifies the name of the variable to be retrieved.

value   FORTRAN string descriptor that receives the blank-filled value of the named variable.

Returns:

A-Register =   status information; see the A-Register Return section at the end of this section.

B-Register =   if the call was successful, the actual length, in characters, of *value* is returned. Otherwise, it equals zero.

Set an Environment Variable to a Given Value

CALL EXEC(39,2,*name*,*value*)

name   FORTRAN string descriptor that specifies the name of the variable to be defined.

value   FORTRAN string descriptor that specifies the value that should be assigned to the named variable.

Returns:

A-Register =   status information; see the A-Register Return section at the end of this section.

B-Register =   undefined.

Delete an Environment Variable

CALL EXEC(39,3,*name*)

name   FORTRAN string descriptor that specifies the name of the variable to be deleted.

Returns:

A-Register =   status information; see the A-Register Return section at the end of this section.

B-Register =   undefined.

Retrieve the Modification Count

CALL EXEC(39,4)

Returns:

A-Register = status information; see the A-Register Return
section at the end of this section.

B-Register = modification count.

A-Register Return

A-Register = status information:

0   Operation successful.

1   Operation successful except the buffer
for *value* provided by the calling program
was not large enough to hold entire value
of the environment variable. The
B-Register contains the length of the
actual value of the environment variable
that resides in the EVB.

2   Named variable not found.

3   Session does not contain an EVB.

4   There is no space left in the EVB to
store the variable. If EXEC(39, 2, ...) is
called to set an environment variable and
this error occurs, the variable retains its
prior value.

2hEV   An EXEC error has occurred.
Remainder of error code is in B-Reg.

2hOP   %ENVRN is not generated into the
system. The B-Register contains 2h39.

There are five possible EXEC 39 errors:

OP39   %ENVRN is not generated into the system.

EV00   Invalid EVB.

EV01   Incorrect number of parameters.

EV02   Illegal parameter value.

EV04   EVB is busy (no-suspend bit set).

10

FMP Routines

File Routines

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.

error   Integer returning negative code (only −231 is possible), or 0 if no error occurs.

dcb   Integer array (at least 16 words) containing Data Control Block (DCB) of a file.

filedescriptor

          Character string unambiguously specifying a single file.

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 Integer output of MaskMatchLevel routine.

destmask   Character string specifying destination mask.

destname   Character string returning full destination file descriptor.

DcbOpen

Purpose: Indicate whether DCB is open.

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

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:	Double integer returning time of last access for a file.
Syntax:	*error* = FmpAccessTime(*filedescriptor*, *time* [, *slink*])
time	Double integer that returns the time of the last access expressed as the number of seconds since Jan 1, 1970.
slink	Optional boolean variable that indicates whether to return the access time of a symbolic link or the file that it references. The possible values are:

> TRUE (negative value)
> Return the access time of the symbolic link file.
>
> FALSE (non-negative value)
> Return the access time of the file referenced by the symbolic link (this is the default).

FmpAppend

Purpose:	Position file of type 3 or above at EOF mark.
Syntax:	error = FmpAppend(*dcb*, *error*)

FmpBitBucket

Purpose: Determine if the type 0 file associated with the specified DCB is LU 0 (the bit bucket).

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

dcb   Integer array containing the DCB for the type 0 file.

bool   Boolean. A flag that is set to TRUE (negative value) if the DCB is open and associated with a type zero file, and the device is LU 0; otherwise, *bool* is set to FALSE (non-negative value).

FmpBuildHierarch

Purpose: Build file descriptor in hierarchical format.

Syntax: *error* = FmpBuildHierarch (*filedescriptor*, *dirpath*, *name*, *typex*, *qual*, *sc*, *type*, *size*, *rl*, *ds*)

filedescriptor 63-character string containing returned file descriptor.

dirpath   Character string (up to 16 characters) naming directory/subdirectory path. *Dirpath* must end with "/", and must have "/" between each of the directories and subdirectories.

name   Character string (up to 16 characters) specifying the file name.

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

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

sc   Integer specifying security code (for FMGR files).

type   Integer specifying FMP file type.

size   Integer specifying file size in blocks.

rl   Integer specifying record length.

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

error   Integer error return. The only possible error is -231 (string too long) which is returned if the string will not fit in the file descriptor. If the call was successful, *error* returns a non-negative value.

FmpBuildName

Purpose:　　　Build file descriptor from given file specifiers.

Syntax:　　　error = FmpBuildName (*filedescriptor*, *name*, *typex*,
　　　　　　　　　　　　　　　　　　　sc, *dir*, *type*, *size*, *rl*, *ds*)

filedescriptor

　　　　　　　63-character string containing returned file descriptor.

name　　　Character string (up to 63 characters) specifying the
　　　　　　　subdirectories, if any, and the file name.

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

sc　　　　Integer specifying security code (for FMGR files).

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

type　　　Integer specifying FMP file type.

size　　　Integer specifying file size in blocks.

rl　　　　Integer specifying record length.

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

error　　　Integer error return. The only possible error is -231
　　　　　　　(string too long), which is returned if the string will not
　　　　　　　fit in the file descriptor.

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* , *rl* , *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.
name	Character string (up to 16 characters) specifying the file name.
qual	Character string mask qualifier (up to 40 characters).

FmpClone Name

Purpose:	Generate program clone names that can be used by FmpRpP rogram.
Syntax:	Call FmpCloneName (*name* , *init*)
name	Character string specifying the program name to be cloned. The specified name is modified by the system and returned to the calling program.
init	Logical indicating whether the current call is the first call to FmpCloneName.

FmpClose

Purpose:	Close file (make it inaccessible).
Syntax:	*error* = FmpClose (*dcb* , *error*)

FmpControl

Purpose:	Issue an I/O device control (EXEC 3) request to LU.
Syntax:	error = FmpControl(*dcb*,*error*,*pram*4*)
dcb	Integer array. Must be associated with a device.
pram1	Integer that is merged with the device LU associated with the given DCB.
pram*3	Up to 3 additional integer parameters (device-dependent).

FmpCopy

Purpose:	Copy a file to another file.
Syntax:	error = FmpCopy(*name1*,*err1*,*name2*,*err2*,*bufr*, *buflen*,*options*)
name1	Character string specifying source file. Can be an LU.
err1	Integer returning errors associated with name1.
name2	Character string specifying destination file. Can be an LU.
err2	Integer returning errors associated with name2.
bufr	Integer buffer (at least 288 words) containing source and destination DCBs and DCB buffers.
buflen	Buffer length, in words (at least 288).
options	Character string selecting copy options, which are:

A = ASCII.
B = Binary.
C = Clear backup bit.
D = Overwrite existing file.
I = Inhibit LU locking of non-interactive devices.
N = Source does not have carriage control.
P = Purge source after copy.
Q = Quiet; do not record access time on source.
S = Preserve directory information (timestamps, protection, and backup bit) of the source file.
T = Truncate destination to length of valid data.
U = Replace duplicate if update time is older.

FmpCreateDir

Purpose: Create directory.

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

 name   Character string specifying name of directory to be created.

 lu   Integer specifying disk LU on which to create directory.

FmpCreateTime

Purpose: Return time that a file was created.

Syntax: *error* = FmpCreateTime(*filedescriptor*, *time* [, *slink*])

 time   Double integer returning time that file was created, expressed in seconds since January 1, 1970.

 slink   Optional boolean variable that indicates whether to return the create time of a symbolic link or the file that it references. The possible values are:

    TRUE (negative value)
     Return the create time of the symbolic link file.

    FALSE (non-negative value)
     Return the create time of the file referenced by the symbolic link (default).

FmpDcbPurge

Purpose: Perform a close and purge on the open file associated with the given DCB.

Syntax: *error* = FmpDcbPurge(*dcb*)

FmpDevice

Purpose: Indicate whether a DCB is associated with a device file.

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

bool   Boolean. TRUE (negative value) if the DCB is associated with a device file; FALSE (non-negative value) if the DCB is associated with a disk file or is closed.

FmpDismount

Purpose: Dismount a volume.

Syntax: *error* = FmpDismount(*lu*)

lu   Integer specifying LU of volume to be dismounted.

FmpEndMask

Purpose: Close files associated with mask search.

Syntax: Call FmpEndMask(*dirdcb*)

dirdcb   Integer array initialized by FmpInitMask.

FmpEof

Purpose: Return position of EOF mark.

Syntax: *error* = FmpEof(*filedescriptor*, *eofpos* [, *slink*])

eofpos   Integer returning current word position of the end of file mark. If the file is open, an incorrect value may be returned.

slink   Optional boolean variable that indicates whether to return the word position of the EOF of a symbolic link or the file that it references. The possible values are:

    TRUE (negative value)
    Return the word position of the EOF of the symbolic link file.

    FALSE (non-negative value)
    Return the word position of the EOF of the file referenced by the symbolic link (default).

FmpError

Purpose: Return error message for FMP error code.

Syntax: `Call FmpError(error,message)`

  message   Character string returning 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   Double integer indicating number of blocks in file.

  size   Integer indicating file size, in one word.

FmpFileName

Purpose: Return full path name of file associated with the specified DCB.

Syntax: *error* = `FmpFileName(dcb,error,filedescriptor)`

FmpHierarchName

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

Syntax: *error* = `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   Returns the file system type; 0 for FMGR, 1 for FMP.

FmpInitMask

Purpose: Initialize file structures for FMP mask calls.

Syntax: *error* = FmpInitMask (*dirdcb*, *error*, *mask*, *diropenname*, *dcblen* [, *msc*])

   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 (372 words minimum).

   msc   System security code. If specified, the routine MaskSecurity and FmpMaskName return the security codes for FMGR files even if the security code was not specified in the original mask.

FmpInteractive

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

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

   bool   Boolean. TRUE (negative value) if the DCB is associated with an interactive device; otherwise, set to FALSE (non-negative value).

FmpIoOptions

Purpose: Return I/O option word.

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

   options   Integer returning 16-bit I/O option word.

FmpIoStatus

Purpose: Return A- and B-Register values.

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

   areg   Integer returning value of A-Register.

   breg   Integer returning value of B-Register.

FmpLastFileName

Purpose: Return file name from the specified file descriptor.

Syntax: `Call FmpLastFileName`(*filedescriptor*,*lastname*)

lastname   Character string returning file name, a portion of
filedescriptor.

FmpList

Purpose: List file to specified LU.

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

lu   Integer specifying output LU.

option   Character string selecting output format and options:
A = ASCII
B = Binary output displayed as octal
C = File has FORTRAN carriage control characters
      in column 1
M = Count lines longer than 80 characters as
      multiple lines for page breaking
Q = Quiet; do not record access time of file
T = Truncate trailing blanks on lines

File types 0, 3, and 4 default to A; all other file types
default to B.

rec1   Double integer specifying first record to be listed.

rec2   Double integer specifying last record to be listed.

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

By default, the listing to an interactive device pauses after printing
one page of output. For sessions that have $LINES defined in their
EVB, the number of lines per page will be the value of $LINES minus
2. When $LINES is not defined in the EVB, the number of lines per
page will be 22.

When FmpList pauses, it prompts you for one of five legal responses.
The responses may be preceded by a number from 1 to 32767 called *n*:

  a or q   Abort the listing
  <space> List another page
   <cr>   List the remainder of the file without pausing
    +   List one more line or skip *n* lines and list 1 more
    p   Set page size to *n* and list another page
    z   Suspend calling program; restart w/ system GO cmnd

FmpListX

Purpose: lists a file to specified file or LU.

Syntax: *error* = FmpListX (*sourcefile*, *destfile*, *options*,
                                    startrec, *endrec*, *buffer*, *maxlength*)

sourcefile   Name of the file to be listed.

destfile   Name of destination listing file.

options   Character string that selects the output format and
                options. The values are as follows:

                A   = ASCII
                B   = Binary output displayed as octal
                C   = File has FORTRAN carriage control characters
                            in column 1
                M = Count lines longer than 80 characters as
                            multiple lines for page breaking
                Q   = Quiet; do not record access time of file
                T   = Truncate trailing blanks on lines

                File types 0, 3, and 4 default to A; all other file types
                default to B.

startrec   First record# to be listed.

endrec   Last record# to be listed.

buffer   Buffer for transporting records between *sourcefile* and
                destfile.

maxlength   Maximum number of bytes that may be contained in
                buffer.

If both *startrec* and *endrec* are set to 0, the entire file is listed.

By default, the listing to an interactive device pauses after printing
one page of output. For sessions that have $LINES defined in their
EVB, the number of lines per page will be the value of $LINES minus
2. When $LINES is not defined in the EVB, the number of lines per
page will be 22.

When FmpList pauses, it prompts you for one of five legal responses.
The responses may be preceded by a number from 1 to 32767 called *n*:

      a or q   Abort the listing
    <space> List another page
      <cr>   List the remainder of the file without pausing
        +   List one more line or skip *n* lines and list 1 more
        p   Set page size to *n* and list another page
        z   Suspend calling program; restart w/ system GO cmnd

FmpLu

Purpose: Return LU number of file or device associated with specified DCB.

Syntax: lu = FmpLu(dcb)

   lu   Integer indicating the LU number of the file or device associated with the specified DCB.

If the DCB is associated with a type 0 file, the value returned in the *lu* parameter is the device LU number. If the DCB is associated with a disk file, the value returned is the LU of the disk on which the file resides. If the specified DCB is not open, a -11 (DCB not open error) is returned.

FmpMakeSLink

Purpose: Create symbolic link file with name specified in *symlink*. The contents of the link is the path specified in *fdesc*.

Syntax: *error* = FmpMakeSLink(dcb, *error*, *fdesc*, *symlink*)

  fdesc   Character string that contains the path name to be used as the contents of the symbolic link being created.

  symlink   Character string that contains the name of the symbolic link file to be created.

Symbolic links may point to either FMP files, FMP directories, or device LU numbers. When creating a link to a file or directory, the *fdesc* must be in hierarchical format. The symbolic link file being created cannot be a FMGR file.

FmpMaskName

Purpose: Build full name for file that matches mask.

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

  dirdcb   Control array initialized by FmpInitMask.

  newname   Character string returning the file descriptor.

  entry   32-word directory entry from FmpNextMask.

  curpath   Character string directory path from FmpNextMask.

FmpMount

Purpose:	Mount a volume.
Syntax:	*error* = FmpMount (*lu*, *flag*, *blks*)
lu	LU of disk volume.
flag	Determines whether to initialize disk before mounting it. The values of flag are:
	0 Do not initialize before mounting.
	1 Initialize if disk does not have a valid directory.
	2 Initialize disk before mounting.
blks	Integer number of blocks to leave free at beginning of volume.

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 (negative value) if there is another entry to be searched, whether or not an error occurred (if error did occur, current entry is invalid). FALSE (non-negative value) if error prevents continuation of search or when search is complete.
dirdcb	Control array initialized by FmpInitMask.
curpath	Returns character string directory path.
entry	32-word array that returns directory entry for each file found.

FmpOpen

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

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

type   Nonnegative integer. Returns file type or error code.

error   An integer that returns a negative code if an error
occurs or returns the type if the call is successful.

options   Character string. Selects file open options from the
following (options can be in any order, uppercase or
lowercase):

Access mode:

R   = open for reading
W = open for writing

File Existence:

C   = create a new file
O   = open an existing file

Miscellaneous:

D   = file descriptor specifies a directory
E   = force LU locking of interactive devices
F   = force type to 1 for unbuffered access
I   = inhibit LU locking of non-interactive devices
L   = open symbolic links
N = file does not contain carriage control
Q = open file quickly, do not record access time
S   = open a shared file
T   = file is temporary
U   = open in update mode
X   = access extents in type 1 or 2 file
n = use UDSP number ($0 \leq n \leq 8$) when
       searching for the file.

buffers   Integer specifying DCB buffer size, in range 1 to 127
(expressed as the number of 128-word buffers in DCB,
in addition to the 16-word file control information).

FmpOpenFiles

Purpose: Indicate open files in a directory.

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

  loc   Integer returning directory position of next open file. Calling program initializes it to 0 to indicate that this is the first call. When all open files in directory are reported, *loc* = −1.

  flag   Integer returning flag value for a file.

FmpOpenScratch

Purpose: An interface to FmpOpen; standardizes the search path used in the creation of scratch files.

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

  type   Same as FmpOpen.

  options   Same as FmpOpen, plus the following:

    Z = pass *filedescriptor* to FmpUniqueName to create a unique scratch file descriptor.

  buffers   Same as FmpOpen.

  nameused   Character string returning the file descriptor of the scratch file that was opened.

FmpOpenTemp

Purpose: An interace to FmpOpen; opens/creates a temporary file.

Syntax: *type* = FmpOpenTemp (*dcb*, *error*, *name*, *options*, *buffers*)

  type   Same as FmpOpen.

  name   Character string specifying characters to be included in the file name. For a program creating more than one file, the *name* string specified for each must be unique.

  options   Same as FmpOpen with the addition that if the T option is not specified, it is automatically added.

  buffers   Same as FmpOpen.

FmpOwner

Purpose: Return name of directory or CI volume owner.

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

dir Character string specifying the name of the directory or number of the CI volume.

owner Character string returning logon name of the user who owns the directory.

FmpPackSize

Purpose: Pack double integer file size into one word.

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

size Integer returning file size in one word.

doublesize Double integer specifying file size.

FmpPagedDevWrite

Purpose: Perform XLUEX (2) write to a device with page breaks for interactive devices.

Syntax: *status* = FmpPagedDevWrite (*cntwd*, *buffer*, *length*, *pageinfo*)

cntwd Two-word integer XLUEX control word specifying LU from 0 to 255 to be written to.

buffer Integer array containing data to be transferred.

length Positive integer indicating number of words, negative integer indicating number of bytes.

pageinfo Five-word array holding paging information for FmpPaginator; see FmpPaginator routine description.

status Integer returning:

0 ready for next line.
1 abort listing.

FmpPagedWrite

Purpose:	Write data to a file using FmpPaginator to break output into screen pages for terminal devices.
Syntax:	*status* = FmpPagedWrite(*dcb*, *error*, *buffer*, *length*, *pageinfo*)
buffer	Character string specifying word-aligned buffer containing data to be transferred.
length	Integer, from 0 to 65534, specifying the number of bytes to write. For a value larger than 32767, set *length* to desired number of bytes minus 65534.
pageinfo	Five-word array holding paging information for FmpPaginator; if first word is zero, the default values are used; see FmpPaginator routine description.
status	Integer returning one of the following:

0   ready for next line.
1   abort listing.

or, negative FMP error code.

FmpPaginator

Purpose:	Prompt for page breaks when FmpList, FmpPagedWrite, and FmpPagedDevWrite routines are used.
Syntax:	*status* = FmpPaginator(*lu*, *paginfo*)
lu	Integer containing LU number, from 0 to 255, to be prompted.
pageinfo	Five-word array holding following information:

word usage
0   page size in lines, or, 0 if default values are desired.
1   lines to print before page break, or, −1 if no paging desired.
2   address of prompt buffer.
3   length of prompt buffer in bytes.
4   flags:
      bit meaning (if set)
      15   do not print current page
      0   use unbuffered I/O

If word 0 is zero, the following default values are given to each word:

word default value

0       $LINES minus 2 if $LINES is defined in EVB, or 22 if $LINES is not defined in the EVB.
1   same value as word 0
2   address of string, "More..."
3   7 characters (length of string)
4   zero (no special flags set)

status   Returns one of the following values:

0   ready for next line.
1   abort listing.
2   continue listing, but skip this line.

Descriptions of valid responses to prompts are as follows (responses may be preceded by *n*, a number from 1 to 32767):

character   action

a or q   Abort the listing
\<space\>   List another page
\<cr\>   List the remainder of the file without pausing
+   List one more line or skip *n* lines and list 1 more
p   Set page size to *n* and list another page
z   Suspend calling program; restart w/ system GO cmnd

FmpParseName

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

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

Parameters are the same as for FmpBuildName except that the file descriptor is specified and the component fields are returned.

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 except that the file descriptor is specified and the component fields are returned.

FmpPosition

Purpose: Return current file position.

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

record   Double integer returning current record number.

position   Double integer returning current internal file position.

FmpPost

Purpose: Post the data in the DCB buffer to the file if the data has been changed.

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

FmpPostEof

Purpose: Post the EOF position of a file and the number of records from the DCB to the directory entry for that file.

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

FmpProtection

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

Syntax: *error* = `FmpProtection`(*filedescriptor*,*owneraccess*, *othersaccess* [,*groupaccess*])

filedescriptor
   Character string specifying the file name, directory name, or CI volume number.

owneraccess   Character string returning owner's access rights.

othersaccess   Character string returning access rights of users other than owner.

groupaccess   Character string returning access rights of members of the associated group.

FmpPurge

Purpose: Purge a file.

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

FmpRawMove

Purpose: Reads or writes data to a disk file starting at a specified
                 internal file position.

Syntax: *length* = FmpRawMove (*dcb*, *error*, *position*,
                                        buffer, *maxlength*, *how*)

    length   Integer that returns the number of words successfully
                 transferred to or from the disk file.

    position   Double integer specifying the desired internal file
                 position.

    buffer   Word-aligned integer buffer that either contains the
                 data to be transferred (*how*=2), or returns the data
                 being transferred (*how*=1).

    maxlength   Integer that contains the number of words to be
                 transferred.

    how   Integer that specifies the direction of the transfer.

                 = 1 Read data from the file into *buffer*.
                 = 2 Write data from *buffer* to the file.

The internal file position after the call is undefined. It is the caller's
responsibility to reset the internal file position after the call.

FmpRead

Purpose:	Read from a file.
Syntax:	*length* = FmpRead(*dcb*, *error*, *buffer*, *maxlength*)
length	Integer returning 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	Integer array returning word-aligned buffer into which data file is to be transferred.
maxlength	Integer containing maximum number of bytes to read. Treated as unsigned single integer from 0 to 65534. Values larger than 32767 are expressed as negative numbers equal to the number of bytes to be transferred minus 65536; for example, 40000 bytes is expressed as −25536 (40000 − 65536 = −25536).
	If an odd number of bytes are transferred, the lower byte of the word containing the last byte is undefined. The requested transfer length can be longer or shorter than the actual length of the record, but the number of bytes read never exceeds *maxlength*.

FmpReadLink

Purpose:	FmpReadLink opens a symbolic link file, returns the contents of the file, and closes the file.
Syntax:	*error* = FmpReadLink(*dcb*, *error*, *symlink*, *fdesc*)
symlink	Character string containing the name of the symbolic link file.
fdesc	Character string that returns the contents of the symbolic link file named in *symlink*.

FmpReadString

Purpose:	Read character string from file.
Syntax:	*length* = FmpReadString(*dcb*, *error*, *string*)
length	Integer returning positive number of bytes transferred, or negative error code.
string	Character string (up to 256 bytes) into which data is transferred.

FmpRecordCount

Purpose: Return number of records in file.

Syntax: *error* = FmpRecordCount (*filedescriptor*, *nrecs* [, *slink*])

nrecs   Double integer returning 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).

slink   Optional boolean variable that indicates whether to return the number of records in a symbolic link file or the file that it references. The possible values are as follows:

> TRUE (negative value)
> Return the number of records in the symbolic link file.

> FALSE (non-negative value)
> Return the number of records in the file referenced by the symbolic link (default).

FmpRecordLen

Purpose: Return length of longest record in file.

Syntax: *error* = FmpRecordLen (*filedescriptor*, *len* [, *slink*])

len   Integer returning length of longest record in file. For file type 3 and above, length of longest record ever written to file even if it has been overwritten.

slink   Optional boolean variable that indicates whether to return the length of the longest record in a symbolic link file or the file that it references. The possible values are as follows:

> TRUE (negative value)
> Return the length of the longest record in the symbolic link file.

> FALSE (non-negative value)
> Return the length of the longest record in the file referenced by the symbolic link (default).

FmpRename

Purpose:	Change file name.
Syntax:	error = FmpRename(*name1*, *err1*, *name2*, *err2*)
name1	Character string specifying name of existing, closed file.
err1	Integer returning error associated file *name1*.
name2	Character string specifying new name.
err2	Integer returning error associated file *name2*.

FmpReportError

Purpose:	Print error message for FMP error code on LU 1.
Syntax:	Call FmpReportError(*error*, *filedescriptor*)
error	Integer specifying the error code whose message is to be written to the terminal.

filedescriptor

Character string specifying name of file to include with error message.

FmpRewind

Purpose:	Position file at its first record.
Syntax:	error = FmpRewind(*dcb*, *error*)

FmpRpProgram

Purpose: Restore program.

Syntax: *error* = `FmpRpProgram`(*filedescriptor*, *rpname*,
                                     options, *error*)

rpname   Character string either specifying or returning program name.

options   Character string containing "C", "D", "P", or both "C" and "P", 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:

- "rpname" is not blank and there is already a program with the specified name RP'd in your session or in the system session. Error −239 is returned in this case.

- there is already a system utility program with the assigned or specified name RP'd in any session, including the system session. Either error −239 or −251 is returned in this case.

- *filedescriptor* does not include a directory path and there is already a program with the assigned name permanently RP'd and dormant in your session or system session. Error −239 returned in this case.

D = (Duplicate) Create prototype ID segment, not program ID segment, for program file.

P = (Permanent) Do not release ID segment when program completes.

FmpRunProgram

Purpose:	Schedule a program.
Syntax:	*error* = FmpRunProgram(*string*,*prams*,*runname* [,*alterstring*])
string	Character string specifying runstring.
prams	Integer array returning RMPAR parameters from program when it completes. If string specifies XQ, these parameters are meaningless.
runname	Character string returning true name used to schedule program.
alterstring	Boolean variable indicating how FMP is to handle string variable. If FALSE, FMP does not alter the string. If TRUE, FMP converts string to uppercase and replaces each group of one or more blanks with a comma.

FmpRwBits

Purpose:	Check string for letters R and W.
Syntax:	*value* = FmpRwBits(*string*)
string	Character string (up to 256 bytes).
value	Integer indicating 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	Double integer specifying number of records in the file plus 1.
eofPos	Double integer specifying current internal file position.
recLen	Integer specifying length of longest record, in words.

FmpSetDirInfo

Purpose: Change directory information.

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

ctime   Double integer specifying create time.

atime   Double integer specifying access time.

utime   Double integer specifying update time.

bbit   Integer specifying back-up bit.

prot   Integer specifying file protection, where:

                 Bit 0=1 General user may write
                 Bit 1=1 General user may read
                 Bit 2=1 Owner may write
                 Bit 3=1 Owner may read
                 Bit 6=   Group may write only if G specified in
                          option string
                 Bit 7=   Group may read only if G specified in
                          option string

                 Any bit set to zero denies the permission associated
                 with that bit.

option   Optional string that determines the interpretation of
                 the *prot* parameter.

                 G = *prot* contains valid group bits. If G is not
                 specified, or if the parameter is not present, the group
                 bits (bits 6 & 7) of *prot* are ignored. In this case, the
                 general user bits (bits 0 and 1) are used for group bits.

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

FmpSetEof

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

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

FmpSetIoOptions

Purpose: Change I/O option word.

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

options   Integer returning 16-bit I/O options word.

FmpSetOwner

Purpose:	Change name of directory owner. You must own directory or be a superuser.
Syntax:	error = FmpSetOwner (*dir*, *err1*, *owner*, *err2*)
dir	Character string specifying directory name.
err1	Integer returning errors associated with *dir*.
owner	Character string specifying name of new owner.
err2	Integer returning errors associated with *owner*.

FmpSetPosition

Purpose:	Change file position.
Syntax:	error = FmpSetPosition (*dcb*, *error*, *record*, *position*)
record	Double integer specifying number of record at which file is to be positioned.
position	Double integer. If positive, desired internal file position; if negative, desired record number.

FmpSetProtection

Purpose:	Change kinds of access available for file, directory, or CI volume. (R = read, W = Write, RW = both).
Syntax:	error = FmpSetProtection (*filedescriptor*, *owneraccess*, *othersaccess* [, *groupaccess*])
filedescriptor	Character string specifying the file name, directory name, or CI volume number.
owneraccess	Character string specifying owner's access rights.
othersaccess	Character string specifying access rights of users other than owner.
groupaccess	Character string specifying access rights of members of the associated group.

FmpSetWord

Purpose: Change file position.

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

position   Integer specifying desired file position:
if positive, internal file position;
if negative, record number.

how   Integer specifying 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

FmpSetWorkingDir

Purpose: Change working directory.

Syntax: error = FmpSetWorkingDir(*directory*)

directory   Character string specifying name of new working
directory.

FmpShortName

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

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

filedescriptor

          Character string returning the name of the file.

FmpSize

Purpose: Return physical file size.

Syntax: *error* = FmpSize (*filedescriptor* , *size* [, *slink*])

  size   Double integer returning physical size in blocks.

  slink   Optional boolean variable that indicates whether to return the physical size of the symbolic link file or the file that it references. The possible values are as follows:

      TRUE (negative value)
        Return physical size of the symbolic link file.

      FALSE (non-negative value)
        Return physical size of the file referenced by the symbolic link (default).

FmpStandardName

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

Syntax: *error* = FmpStandardName (*filedescriptor*)

FmpTruncate

Purpose: Truncate file.

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

  blocks   Double integer specifying minimum number of blocks to which file will be truncated.

FmpUdspEntry

Purpose: Return the directory name for the specified entry and User-Definable Directory Search Path (UDSP).

Syntax: *error* = FmpUdspEntry(*udspnum*, *entnum*, *dirname*,
                                              error)

udspnum   Integer specifying the UDSP number.

entnum   Integer specifying the entry for the UDSP number.

dirname   Character string returning the directory name for the specified entry in the specified UDSP.

error   Integer returning a negative code if an error occurs or zero if no error occurs.

        0   No error occurred
      −1   Not under session control
      −2   UDSP tables not set up correctly
  −247   If the entry is undefined, or if *udspnum* and *entnum* are out of bounds with the definition for the session.

FmpUdspInfo

Purpose: Returns the current User-Definable Directory Search Path (UDSP) information for your session.

Syntax: *error* = FmpUdspInfo(*udsps*, *depth*, *next*, *error*)

udsps   Integer returning the number of UDSPs defined for the current session.

depth   Integer returning the UDSP depth defined for the current session.

next   Integer returning the next available UDSP. *next* is set to zero if all UDSPs are defined.

error   Integer returning one of the following values:

        0   No error occurred
     −1   Not under session control
     −2   UDSP tables not set up correctly

FmpUniqueName

Purpose: Create and return unique file name.

Syntax: Call FmpUniqueName (*prefix, uniquename*)

 prefix Character string specifying prefix for unique file name.

 uniquename Character string returning the generated file name.

FmpUnPurge

Purpose: Restore a file.

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

FmpUpdateTime

Purpose: Return last update time for a file.

Syntax: *error* = FmpUpdateTime (*filedescriptor, time* [*, slink*])

 time Double integer returning time of last update, expressed in seconds since Jan 1, 1970.

 slink Optional boolean variable that indicates whether to return the time of the last update of the symbolic link file or the file that it references. The possible values are as follows:

> TRUE (negative value)
> Return the time of the last update of the symbolic link file.
>
> FALSE (non-negative value)
> Return the time of the last update of the file referenced by the symbolic link (default).

FmpWorkingDir

Purpose: Return current working directory.

Syntax: error = FmpWorkingDir (*directory* [, *format*])

   directory   Character string returning name of current working
                directory.

   format   Optional integer parameter that defines the format of
                the directory string being returned. Possible values for
                format and their definitions are:

        0   (default) if a working directory is a global
           directory, it is returned in the trailing directory
           format ("::dir"); otherwise, the working
           directory is returned in hierarchical format with
           no trailing slash.

        1   the working directory is returned in hierarchical
           format with no trailing slash.

        2   the working directory is returned in hierarchical
           format with a trailing slash.

FmpWrite

Purpose: Write to a file.

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

   length   Integer returning 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. For values
                larger than 32767, set *maxlength* to the desired
                maximum number of bytes minus 65536; for example,
                40000 bytes is expressed as −25536 (40000 − 65536 =
                −25536).

FmpWriteString

Purpose: Write character string to file.

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

length   Integer returning length of record written to file, or negative error code.

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

MaskDiscLu

Purpose: Return the disk lu of the last file returned by FmpNextMask

Syntax: *diskLu* = MaskDiscLu(*dirdcb*)

diskLu   Integer returning the disk LU.

dirdcb   Integer array, initialized by FmpInitMask.

MaskIsDS

Purpose: MaskIsDS is a logical function that determines if masking is searching a remote file system.

Syntax: *bool* = MaskIsDS(*dirdcb*[, *dsinfo*])

bool   Boolean variable that returns TRUE (negative value) if masking is searching a remote file system; otherwise, returns FALSE (non-negative value).

dirdcb   Control array, initialized by FmpInitMask.

dsinfo   Optional character string that returns the DS information of the mask. For example, the remote user account name, node name, or both, along with the required delimiters are returned, as in ">27", ">SYS3", "[USER]", and ">SYS3[USER/PASSWORD]".

MaskMatchLevel

Purpose: Return directory level of last file matched.

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

matchlevel   Integer returning the directory level containing the last file that was matched.

dirdcb   Integer array initialized by FmpInitMask.

MaskOldFile

Purpose: Determine if file is a FMGR file.

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

bool   Boolean. Returns TRUE if last file returned by FmpNextMask is a FMGR file; otherwise, returns FALSE.

dirdcb   Integer array initialized by FmpInitMask.

MaskOpenId

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

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

openid   Integer returning D.RTR open flag of last file returned by FmpNextmask; 0 if file is closed.

dirdcb   Integer array initialized by FmpInitMask.

MaskOwnerIds

Purpose:	Return the owner and group IDs for the last file returned by FmpInitMask.
Syntax:	`Call MaskOwnerIDs`(*dirdcb*,*ownerid*,*groupid*)
dirdcb	Integer array; initialized by FmpInitMask.
ownerid	Integer returning ID number of the file owner.
groupid	Integer returning ID number of the associated group of the file.

MaskSecurity

Purpose:	Return security code of last FMGR file returned by FmpNextMask.
Syntax:	*seccode* = `MaskSecurity`(dirdcb)
seccode	Integer returning security code of last file returned by FmpNextMask if file was an FMGR file; 0 for FMP file.
dirdcb	Integer array initialized by FmpInitMask.

WildCardMask

Purpose:	Check for wildcard characters in mask.
Syntax:	*wild* = `WildCardMask`(*mask*)
wild	Boolean. Returns TRUE if *mask* refers to more than one file; FALSE otherwise.
mask	Character string containing mask to be checked.

Using the FMP Routines with DS

DsCloseCon

Purpose: Close connection set up by DsOpenCon.

Syntax: *error* = `DsCloseCon(`*conn*`)`

  conn   Integer specifying connection number to be closed.

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   Integer specifying connection number of system.

  word   Integer returning first word of DCB.

DsDiscInfo

Purpose: Return number of tracks and blocks per track for
specified disk volume.

Syntax: *error* = `DsDiscInfo(`*conn*`,`*lu*`,`*ntracks*`,`*bpert*`)`

  conn   Integer specifying the connection number of system
containing disk.

  lu   Integer specifying LU of disk volume.

  ntracks Integer returning number of tracks on disk volume.

  bpert   Integer returning number of blocks per track on disk.

DsDiscRead

Purpose:	Read disk on system specified by connection number.
Syntax:	*error* = DsDiscRead(*conn*, *buf*, *len*, *track*, *sector*)
conn	Integer specifying connection number of system. Must have been set by DsSetDcbWord.
buf	Integer array. Buffer to hold data read from disk.
len	Integer specifying number of characters to read (up to 4096).
track	Integer specifying track from which to read.
sector	Integer specifying 64-word sector from which to read (even number).

DsFstat

Purpose:	Perform FSTAT call for specified system.
Syntax:	*error* = DsFstat(*conn*, *buf*, *len* [, *format* [, *iop*]])
conn	Integer specifying connection number of system.
buf	Integer array returning status of cartridges. At least 256 words.
len	Integer specifying length of buffer in words.
format	Same as for FSTAT (used only if remote node is RTE-6/VM system).
iop	Same as for FSTAT (used only if remote node is RTE-6/VM system).

DsNodeNumber

Purpose: Return node number associated with specified file.

Syntax: *node* = DsNodeNumber (*filedescriptor*)

   node   Integer returning the number of the node associated with the specified file. A zero is returned if the file is not remote.

DsOpenCon

Purpose: Open connection to remote user account/node.

Syntax: *error* = DsOpenCon (*string*, *conn*)

  string   Character string specifying remote user account name, node name, or both, along with required delimiters. Must not contain a file name, only DS information.

  conn   Integer returning connection number.

DsSetDcbWord

Purpose: Change first word of DCB so that DsDiskRead works.

Syntax: *error* = DsSetDcbWord (*conn*, *word*)

  conn   Integer returning connection number of system.

  word   Integer specifying word to be changed.

FMGR File Routines

The subroutines in this subsection handle files in FMGR format. The following parameters have the same meaning in each subroutine.

idcb   Data Control Block (DCB); an array of 144 + ($n*128$) words, where n is positive or zero.

ierr   Integer returning negative error code; 0 or positive for no error.

name   File name in 3-word array.

isize File size in 2-word array (maximum 16384 blocks). Word is normally 16 bits; 32 bits for extended-range subroutines. First word contains number of blocks; -1 allocates rest of cartridge to file. Second word, used for type 2 file only, contains record length in words.

itype Integer returning file type, in range 1 to 32767.

isecu Integer returning security code, in range -32768 to 32767. Positive value write-protects only, negative value read- and write-protects. Default is 0 (unprotected).

icr   Integer cartridge reference. Positive value is cartridge reference number, negative value is LU. Default is first cartridge with room for file.

idcbs Integer indicating DCB buffer size, if larger than 128. Default is 144 words for buffer and control words, regardless of *idcb* dimensions.

APOSN and EAPOS

Purpose: Position disk file to specific records, possibly determined by last LOCF call. EAPOS is extended-range version of APOSN.

Syntax: CALL APOSN (*idcb*, *ierr*, *irec* [, *irb* [, *ioff*]])
CALL EAPOS (*idcb*, *ierr*, *irec* [, *irb* [, *ioff*]])

    irec   Integer indicating number of next record, in range 1 to 32767 for APOSN; 1 to 2147483647 for EAPOS. Possibly set by last LOCF call.

    irb   Integer indicating number of next block, 16383 or less. Possibly set by last LOCF call. Required with variable-length records.

    ioff   Integer indicating number of next word in DCB, in range 0 to 127. Possibly set by last LOCF call. Required with variable-length records.

CLOSE and ECLOS

Purpose: Close a file, truncating it if desired. ECLOS is extended-range version of CLOSE.

Syntax: CALL CLOSE (*idcb* [, *ierr* [, *itrun*]])
CALL ECLOS (*idcb* [, *ierr* [, *itrun*]])

    itrun   Integer containing number of blocks to be deleted from file, less than or equal to 16384 for CLOSE; less than or equal to 1073741824 for ECLOS. If negative, only extents are truncated. Default is no truncation.

CRDC

Purpose: Dismount cartridge from system.

Syntax: *ierr* = CRDC (*icr*)

    icr   Integer containing positive Cartridge Reference Number or negative LU of mounted cartridge.

CREAT and ECREA

Purpose: Create a file and open it exclusively to the program creating it. If it is a disk file, allocate disk space for the data. ECREA is extended-range version of CREAT.

Syntax: CALL CREAT (*idcb*, *ierr*, *name*, *isize*, *itype*
[, *isecu* [, *icr*] [, *idcbs*]]])

CALL ECREA (*idcb*, *ierr*, *name*, *isize*, *itype*
[, *isecu* [, *icr*] [, *idcbs* [, *jsize*]]]])

ierr   If no error, number of 64-word sectors in created file.

jsize   Integer returning actual file size in sectors if ECREA is successful (32-bit value).

CRETS

Purpose: Create temporary or scratch disk file and open it exclusively to the program creating it.

Syntax: CALL CRETS (*idcb*, *ierr*, *num*, *name*
[, *isize* [, *itype* [, *isecu* [, *icr* [, *idcbs* [, *jsize*]]]]]])

num   Integer specifying scratch file number between 0 and 99.

jsize   Integer returning actual file size if CRETS is successful (32-bit value).

CRMC

Purpose: Mount cartridge to system.

Syntax: *ierr* = CRMC (*lu* [, *lstrk*])

lu   Integer indicating nonnegative LU of unmounted cartridge.

lstrk   Integer indicating last track on cartridge available for FMP use.

FCONT

Purpose: Controls I/O on open type 0 file. Same functions as EXEC I/O control call (EXEC 3).

Syntax: CALL FCONT (*idcb*, *ierr*, *icnwd*, *iprm1*, *iprm2*, *iprm3*, *iprm4*)

icnwd   Control word.

iprm1...iprm4

    Device-dependent parameters.

FSTAT

Purpose: Return array containing system cartridge list.

Syntax: *ierr* = FSTAT (*istat* [, *ilen*])

istat   Array containing returned cartridge list.

ilen   Integer indicating length of *istat* in words (default is 125).

IDCBS

Purpose: Return number of words of Data Control Block that the File Management Package uses for data transfer and file control.

Syntax: *isize* = IDCBS (*idcb*)

isize   Integer returning DCB buffer size.

LOCF and ELOCF

Purpose: Retrieve status and location of open file. ELOCF is extended-range version of LOCF.

Syntax: CALL LOCF (*idcb*, *ierr*, *irec* [, *irb* [, *ioff* [, *jsec* [, *jlu* [, *jty* [, *jrec*]]]]]])

CALL ELOCF (*idcb*, *ierr*, *irec* [, *irb* [, *ioff* [, *jsec* [, *jlu* [, *jty* [, *jrec*]]]]]])

irec Integer returning next sequential record, in range 1 to 32767 for LOCF; 1 to 1073741824 for ELOCF.

irb Integer returning next block, unless file is type 0. Includes extents. In range 0 to 16383 for LOCF, 0 to 1073741824 for ELOCF.

ioff Integer returning next word in block, unless file is type 0. In range 0 through 127.

jsec Integer returning number of sectors in file at creation, unless file is type 0. Even number in range 2 to 32766 for LOCF; 2 to 1073741824 for ELOCF.

jlu Integer returning file LU.

jty Integer returning file type.

jrec Integer returning record size of type 1 or 2 file, read/write code for type 0 file; does not apply to file of type 3 or higher.

NAMF

Purpose:	Rename existing file.
Syntax:	CALL NAMF (*idcb*, *ierr*, *name*, *nname* [, *isecu* [, *icr*]])
name	Old file name in three-word array.
nname	New file name in three-word array.
isecu	Integer returning security code. Can be omitted if file was created with 0 or no security code.
icr	Integer returning cartridge reference in range 0 through 32767; if 0 or omitted, the first file with old file name and matching security code is renamed.

OPEN

Purpose:	Open existing file for read and/or write access.
Syntax:	CALL OPEN (*idcb*, *ierr*, *name* [, *ioptn* [, *isecu* [, *icr* [, *idcbs*]]]])
ierr	Integer returning file type if open succeeds.
ioptn	Open options, specified by octal value representing one-word variable after bits have been appropriately set:

E (bit 0)	= 0:	File opened exclusively to calling program; locked if type 0.
	= 1:	File can be shared by up to seven programs.
U (bit 1)	= 0:	File opened for standard (not update) write.
	= 1:	File opened for update.
T (bit 2)	= 0:	File opened for standard (not update) write.
	= 1:	Force file type to 1.
EX (bit 5)	=	Permit extents on type 1 and 2 files.

Bit F is ignored for all but type 0 files:

F (bit 3)   = 0: Use function code defined at creation.

            = 1: Use function code defined in bits 6-10 of *ioptn*.

Bits 6-10 correspond exactly to function code used for read or write call (EXEC 1 or 2).

If omitted or 0, file is opened exclusively to the calling program, for standard sequential output, with the file type or function code assigned at creation.

OPENF

Purpose: Open device or existing file for read or write access. Creates DCB to allow access to device without creating type 0 file.

Syntax: CALL OPENF (*idcb*, *ierr*, *name* [, *ioptn* [, *isecu* [, *icr*
                                                      [, *idcbs*]]])

ierr   Integer returning file type if open succeeds.

ioptn   Open options, specified by octal value representing one-word variable after bits have been appropriately set (see OPEN command). If omitted or 0, file is opened exclusively to the calling program, for standard sequential output, with the file type or function code assigned at creation.

POSNT and EPOSN

Purpose:	Position file relative to current position or to specified record. EPOSN is extended-range version of POSNT.
Syntax:	CALL POSNT (*idcb*, *ierr*, *nur*, *ir*)
	CALL EPOSN (*idcb*, *ierr*, *nur*, *ir*)
nur	Integer specifying number of records to move forward (if positive) or backward (if negative). If nonzero, *ir* is specified, *nur* is the number of the record where the file is to be positioned. For POSNT, absolute value of *nur* cannot exceed 32767; for EPOSN, it cannot exceed 1073741824.
ir	Integer indicating that *nur* is the number of the record where the file is to be positioned.

POST

Purpose:	Write Data Control Block buffer to disk file.
Syntax:	CALL POST (*idcb* [, *ierr*])
ierr	Can be omitted only if A-Register is tested for errors.

PURGE

Purpose:	Remove file and extents from system.
Syntax:	CALL PURGE (*idcb*, *ierr*, *name* [, *isecu* [, *icr*]])

READDF and EREAD

Purpose: Transfer record (or part of record) from open file to user buffer. EREAD is extended-range version of READF.

Syntax: CALL READF (*idcb*, *ierr*, *ibuf* [, *il* [, *len* [, *num*]]])
           CALL EREAD (*idcb*, *ierr*, *ibuf* [, *il* [, *len* [, *num*]]])

  ibuf   User buffer (of length *il* if *il* is specified).

  il   Integer specifying number of words to be read. Required for type 0 file; default is one for other file types.

  len   Integer specifying actual number of words read; -1 if end-of-file is read.

  num   Integer. If positive, number of record to be read; if negative, number of records to backspace. If omitted, record at current position is read. In range -32768 to 32767 for READF; -2147483648 to 2147483647 for EREAD. Applies to file types 1 and 2 only.

RWNDF

Purpose: Rewind nondisk file or position disk file at its first record.

Syntax: CALL RWNDF (*idcb* [, *ierr*])

  ierr   Can be omitted only if A-Register is to be checked for errors.

WRITF and EWRIT

Purpose: Transfer record from user buffer to open file. EWRIT is extended-range version of WRITF.

Syntax: CALL WRITF (*idcb* , *ierr* , *ibuf* [, *il* [, *num*]])
                CALL EWRIT (*idcb* , *ierr* , *ibuf* [, *il* [, *num*]])

  ibuf   User buffer; array containing record to be written.

  il   Integer specifying number of words to be written. Default is one record for file types 1 and 2; a zero-length record for other file types.

  num   Integer. If positive, number of record to be written; if negative, number of records to backspace. If omitted or 0, record is written to current file position. In range -32768 to 32767 for WRITF; -2147483648 to 2147483647 for EWRIT. Applies only to file types 1 and 2.

11

VMA/EMA Routines

General Purpose VMA/EMA Subroutines

EIOSZ

Purpose: Determine maximum length of transfer. (Provided for RTE-6/VM compatibility.)

Syntax: CALL EIOSZ (isize)
              isize = EIOSZ ( )

isize   Always returns 100000B.

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), excluding spillover page.

imseg   Starting logical page of MSEG.

iws   Working set page size. For EMA, same as nema.

Returns:

   A-Register

      = 0 if normal return
      = −1 if error occurred

   B-Register

      = 0 if Normal Model program
      = 1 if Large Model program
      = 2 if Extended Model program

LKEMA/ULEMA

Purpose: Lock/unlock a shareable EMA partition.

Syntax: CALL LKEMA   lock partition
              CALL ULEMA   unlock partition

LOCKVMA, LOCKVMABUF, LOCKVMA2BUF

Purpose: These routines lock and unlock pages and buffers in
              virtual memory. LOCKVMA locks from 1 through 64
              pages and is useful for locking two or more
              noncontiguous areas into memory at once.
              LOCKVMABUF locks a single VMA buffer area into
              memory. LOCKVMA2BUF locks two VMA buffer
              areas into memory.

Syntax: error = LOCKVMA(ipages, inumpg)
              error = LOCKVMABUF(ibuf, ilen)
              error = LOCKVMA2BUF (ibuf1, ilen1, ibuf2, ilen2)

error   Integer returning negative code, or 0 if successful.

              −82 Specified page numbers or buffer is out of bounds.

              −89 Ipages not in range 0 through 64 inclusive, or
                  specified buffer is too large.

ipages   Name of integer array to be locked. The array can
              contain a maximum of 64 virtual pages.

inumpg   Number of pages specified in ipages. Can be set to a
              value between 1 and 64, inclusive, to lock pages. A
              value of zero unlocks all pages.

ibuf, ibuf1, and ibuf2
              Two-word virtual address of the VMA buffers to be
              locked into memory.

ilen, ilen1, and ilen2
              One-word integer specifying the length of ibuff, ibuf1,
              and ibuf2, respectively, in words (positive value) or
              bytes (negative value).

Calling LOCKVMA releases all pages previously locked by either
LOCKVMABUF or LOCKVMA2BUF. All calls to any of these three
routines unlock all previously locked pages.

Any error return unlocks any previously locked pages.

VMAIO

Purpose: Perform VMA or EMA data transfers to or from a device.

Syntax: For non-class I/O (ECODE is 1 or 2):

CALL VMAIO
(ecode,cntrl,ibuff,ilen[,pram3[,pram4,class]])

For class I/O (ECODE is 17, 18, or 20):

CALL
VMAIO(ecode,cntrl,ibuff,ilen,pram3,pram4,class
[,uv[,keynum]])

For a class GET (ECODE is 21):

CALL VMAIO (ecode,class,ibuff,ilen[,rtn1[,rtn2[,rtn3
[,uv]]]])

cntrl   Two-word quantity with the following format:

WORD1		WORD2	
Bits 0−7	LU number	Bits 0−5	Reserved
Bits 8−11	Reserved	Bit 6	BI bit
Bit 12	WT bit	Bit 7	Reserved
Bit 13	Reserved	Bit 8	EC bit
Bit 14	OS bit	Bit 9	Reserved
Bit 15	OV bit	Bit 10	TR bit
		Bit 11	Reserved
		Bit 12	Z bit
		Bit 13	UE bit
		Bit 14	NB bit
		Bit 15	BB bit

ibuff   Two-word integer that specifies the VMA/EMA word offset of the buffer.

ilen   Integer that specifies the length of the buffer in words (positive value) or bytes (negative value).

pram3   Optional parameter or buffer, as in the EXEC 1 or 2 call.

pram4   Optional parameter of buffer length.

class	Integer that specifies the class number.
uv	User-defined variable that is returned from a previous read or write.
keynum	Integer that specifies the key number of the locked LU.
rtn1	Corresponds to pram3 for a read or write call.
rtn2	Corresponds to pram4 for a read or write call.
rtn3	Request code that is passed to the driver on an initial read or write as follows:

1   Call was 17 or 20 (read or write/read)
2   Call was 18 (write)

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.

Returns:

A-Register
=   0 if Normal Model program
=   1 if Large Model program
=   2 if Extended Model program

VMA Backing Store File Subroutines

Default file names:

　　　　　　a)　on the FMGR cartridge: nnncVM
　　　　　　b)　on the CI volume: VMcnnn.VMA

where:

　　　　　　nnn　= ID segment number of the VMA program
　　　　　　c　　= CPU number

The default backing store is a type 2 file in 256 block increments.

If the SC BOOTEX command specified a zero, the default backing store file goes either on the /SRATCH/ directory (if it exists) or on the current working directory.

If the SC BOOTEX command specified other than zero, then the backing store file goes on the specified FMGR cartridge.

VMACLOSE

Purpose:　　　Close the VMA backing store file.

Syntax:　　　**CALL VMACLOSE**

VMAOPEN

Purpose: Create or open backing store file.

Syntax: CALL VMAOPEN (ierr,name,ioptn)

   ierr   Integer returning error code (0 indicates successful call).

   name   Character string indicating file pathname (63 or fewer characters), DS information is ignored; name may be blank.

   ioptn   Character string indicating file options; character string list of one-letter options (upper or lowercase) selected from the following set:

        R   =   Open for reading
        W   =   Open for writing
        O   =   OK to open an existing backing store file
        C   =   OK to create a new backing store file
        S   =   Open shared
        U   =   Open in update mode
        X   =   OK to access and/or create extents
        T   =   File is temporary
        V   =   Defer create/open until required

VMAPOST

Purpose: Post working set to disk.

Syntax: CALL VMAPOST

VMAPURGE

Purpose: Purge VMA backing store file.

Syntax: CALL VMAPURGE

VMAREAD

Purpose: Read data from a file to a VMA/EMA.

Syntax: ilen = VMAREAD (idcb,ierr,iarray,idl)

idcb   Data Control Block (DCB). Previously specifed
        integer array of 144+n words; n is zero or positive.

ierr   Integer returning FMP error code. Zero indicates
        successful call.

iarray   Two-word address specifying data transfer destination
        start address in VMA/EMA.

idl   Integer specifying data length requested, in bytes.

ilen   Integer returning actual data length read (in bytes) or
        negative error code. Also negative if more than 32767
        bytes were read.

VMAWRITE

Purpose: Write data from VMA/EMA to a file.

Syntax: ilen = VMAWRITE (idcb,ierr,iarray,idl)

idcb   Data Control Block (DCB). Previously specified
        integer array of 144 +n words; n is zero or positive.

ierr   Integer returning FMP error code. Zero indicates
        successful call.

iarray   Two-word address specifying data transfer destination
        start address in VMA/EMA (positive).

idl   Integer specifying data length requested, in bytes.

ilen   Integer returning actual number of bytes transferred,
        or negative error code. Also negative if more than
        32767 bytes are transferred.

FMGR VMA File Routines

CLSVM

Purpose: Close the VMA backing store file.

Syntax: Call CLSVM

CREVM

Purpose: Create a VMA backing store file.

Syntax: CALL CREVM (name,ierr,ioptn,isc,icr)

name   Three-word array returning file name.

ierr   Integer returning error code. Zero indicates successful
call.

ioptn   File options:

Bit 0 = 0   Ignore name and bit 1.
Bit 0 = 1   Create non-scratch file for backing store
file.
Bit 1 = 1   Open file if create fails due to duplicate
file error.
Bit 2 = 1   Defer file create until working set must
be written to file.
Bit 3 = 1   Do not create or address file extents.

isc   Integer specifying file security code.

icr   Integer specifying file Cartridge Reference Number
(CRN).

OPNVM

Purpose:	Open a VMA backing store file.
Syntax:	CALL OPNVM (name,ierr,ioptn,isc,icr)
name	Three-word array returning file name.
ierr	Integer returning error code. Zero indicates successful call.
ioptn	File options:

Bit 0 = 1 Open file for nonexclusive use.
Bit 1 = 1 Open file for update.
Bit 2 = 1 Defer file open until required.
Bit 3 = 1 Do not address or create file extents.
Bit 4 = 1 Open disk file for read-only access

| isc | Integer specifying file security code. |
| icr | Integer specifying file Cartridge Reference Number (CRN). |

PSTVM

| *Purpose:* | Post working set to disk. |
| *Syntax:* | CALL PSTVM |

PURVM

| *Purpose:* | Purge VMA backing store file. |
| *Syntax:* | CALL PURVM |

VREAD

Purpose:	Read data from a file to a VMA/EMA.
Syntax:	CALL VREAD (idcb,ierr,iarray,idl[,ilen[,inum]])

idcb Data Control Block (DCB). Previously specified integer array of 144+n words; n is zero or positive.

ierr Integer returning error code. Zero indicates successful call.

iarray Two-word integer representing data transfer destination start address in VMA/EMA (positive).

idl Integer specifying data length requested in words (positive).

ilen Integer indicating actual data length read.

inum Integer specifying positive offset (in records) from beginning of file or negative offset from current position. Default = current position.

VWRIT

Purpose:	Write data from VMA/EMA to a file.
Syntax:	CALL VWRIT (idcb,ierr,iarray,idl[,inum])

idcb Data Control Block (DCB). Previously specified integer array of 144+n words; n is zero or positive.

ierr Integer returning error code. Zero indicates successful call.

iarray Two-word integer representing data transfer destination start address in VMA/EMA (positive).

idl Integer specifying data length requested, in words (positive).

inum Integer specifying positive record number or negative number of records to backspace (file types 1 or 2 only). Default is record at current position.

VMA/EMA Mapping Management Subroutines

.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:

B-Register contains logical address of element.
A-Register is meaningless.

Table:

> Number of Dimensions
>
> $-L(n)$
> $d(n-1)$
> $-L(n-2)$
> $d(n-2)$
> ⋮
> $-L(2)$
> $d(1)$
> $-L(1)$
> Number of words per element
> Offset word 1 (bits $15-0$)
> Offset word 2 (bits $31-16$)

where:

$L(i)$	is the lower bound of the ith dimension.
$d(i)$	is the number of elements in the ith dimension.

.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.
```

Normal return:

> All VMA/EMA pages are mapped into logical memory and B-Register equals the logical address of the starting page of MSEG.

.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 the element.
A-Register is undefined.

Table:

```
        DEC                Number of dimensions.
        DEC D(n-1)         Number of elements in the (n-1)
                           dimension.
        DEC D(n-2)         .
          :                  :
        DEC D(1)           Number of elements in the first
                           dimension.
        DEC                Number words per element.
        BSS 2              Offset word in double integer format
                           (most significant word first, least
                           significant word last).
```

.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:

                B-Register contains logical address of the element.
                A-Register is undefined.

Table:

                DEC   Number of dimensions.
                DEC D(n−1)   Number of elements in the (n−1)
                                dimension.
                DEC D(n−2)   .
                :   :
                DEC D(1)   Number of elements in the first
                                dimension.
                DEC   Number of words per element.
                BSS 2   Offset word in double integer format
                                (most significant word first, least
                                significant word last).

.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.

Table:

DEC	Number of dimensions.
DEC D(n-1)	Number of elements in the (n-1) dimension (High Bits).
DEC D(n-1)	Number of elements in the (n-1) dimension (Low Bits).
:	:
DEC D(1)	Number of elements in the first dimension (High Bits).
DEC D(1)	Number of elements in the first dimension (Low Bits)
DEC	Number of words per element.
BSS 2	Offset word in double integer format.

.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).

Table:

DEC	Number of dimensions.
DEC D(n−1)	Number of elements in the (n−1) dimension (High Bits).
DEC D(n−1)	Number of elements in the (n−1) dimension (Low Bits).
:	:
DEC D(1)	Number of elements in the first dimension (High Bits).
DEC D(1)	Number of elements in the first dimension (Low Bits).
DEC	Number of words per element
BSS 2	Offset word in double integer format.

.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.

MMAP Subroutine

Purpose: Map consecutive pages of EMA or VMA into logical memory.

FORTRAN 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.

Models of EMA/VMA

There are three "models" of EMA/VMA for programs to use:

1. The Normal EMA/VMA model is used by programs that do not need to access more than one shareable EMA area, nor to access a shareable EMA in conjunction with a local EMA or VMA, nor to use an extended EMA or VMA working set size. This model is the most commonly used.

2. The Large EMA/VMA model is used by programs that need to access more than one shareable EMA area, or to access a shareable EMA and a local EMA or VMA.

3. The Extended EMA/VMA model is used by programs running on an A990 Computer that need to access an EMA or VMA working set that is larger than 1,022 pages. This model also offers all the features of the Large model: multiple shareable EMAs, and mixed shareable EMA and local EMA/VMA. Programs that use this model will execute correctly only on an A990 Computer.

12

●

System Tables

ASCII Character Set in the Two-Byte Word

Effect of Control Key*

◄ 000–037B ► ◄ 040–077B ► ◄ 100–137B ► ◄ 140–177B ►

b_4 b_3 b_2 b_1	row	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	\	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

BITS / column / row

0

32 CONTROL CODES

Unshifted Lowercase

◄— 64 CHARACTER SET —►
◄— 96 CHARACTER SET —►
◄— 128 CHARACTER SET —►

Example: The representation for the character "K" (column 4, row 11) is:

b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1

BINARY   0 1   001   011

OCTAL   1   1   3

* Depressing the Control key while typing an uppercase letter produces the corresponding control code on most terminals. For example, Control–H is a backspace.

ASCII Character Set and Octal Codes

Effect of Control Key*

	00x	01x	02x	03x	04x	05x	06x	07x	10x	11x	12x	13x	14x	15x	16x	17x
xx0	Nul	Bs	Dle	Can	Sp	(	0	8	@	H	P	X	`	h	p	x
xx1	Soh	Ht	Dc1	Em	!	)	1	9	A	I	Q	Y	a	i	q	y
xx2	Stx	Lf	Dc2	Sub	"	*	2	:	B	J	R	Z	b	j	r	z
xx3	Etx	Vt	Dc3	Esc	#	+	3	;	C	K	S	[	c	k	s	{
xx4	Eot	Ff	Dc4	Fs	$	,	4	<	D	L	T	\	d	l	t	|
xx5	Enq	Cr	Nak	Gs	%	–	5	=	E	M	U	]	e	m	u	}
xx6	Ack	So	Syn	Rs	&	.	6	>	F	N	V	^	f	n	v	~
xx7	Bel	Si	Etb	Us	'	/	7	?	G	O	W	_	g	o	w	Del

* Depressing the Control Key while typing an uppercase letter produces the corresponding control code on most terminals. For example, Control-H is a backspace.

ASCII Character Set (continued)

Decimal Value	Octal Values		Mnemonic	Graphic	Meaning
	Left Byte	Right Byte			
0	000000	000000	NUL	N_U	Null
1	000400	000001	SOH	S_H	Start of Heading
2	001000	000002	STX	S_X	Start of Text
3	001400	000003	ETX	E_X	End of Text
4	002000	000004	EOT	E_T	End of Transmission
5	002400	000005	ENQ	E_Q	Enquiry
6	003000	000006	ACK	A_K	Acknowledge
7	003400	000007	BEL		000007
8	004000	000010	BS	B_S	Backspace
9	004400	000011	HT	H_T	Horizontal Tabulation
10	005000	000012	LF	L_F	Line Feed
11	005400	000013	VT	V_T	Vertical Tabulation
12	006000	000014	FF	F_F	Form Feed
13	006400	000015	CR	C_R	Carriage Return
14	007000	000016	SO	S_O	Shift Out } Alternate Character Set
15	007400	000017	SI	S_I	Shift In
16	010000	000020	DLE	D_L	Data Link Escape
17	010400	000021	DC1	D_1	Device Control 1 (X−ON)
18	011000	000022	DC2	D_2	Device Control 2 (TAPE)
19	011400	000023	DC3	D_3	Device Control 3 (X−OFF)
20	012000	000024	DC4	D_4	Device Control 4 (TAPE)
21	012400	000025	NAK	N_K	Negative Acknowledge
22	013000	000026	SYN	S_Y	Synchronous Idle
23	013400	000027	ETB	E_B	End of Transmission Block
24	014000	000030	CAN	C_N	Cancel
25	014400	000031	EM	E_M	End of Medium
26	015000	000032	SUB	S_B	Substitute
27	015400	000033	ESC	E_C	Escape [2]
26	016000	000034	FS	F_S	File Separator
29	016400	000035	GS	G_S	Group Separator
30	017000	000036	RS	R_S	000036
31	017400	000037	US	U_S	000037
127	077400	000177	DEL	▇	Delete, Rubout

ASCII Character Set (continued)

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
32	020000	000040	.	Space, blank
33	020400	000041	!	Exclamation Point
34	021000	000042	"	Quotation Mark
35	021400	000043	#	Number Sign, Pound Sign
36	022000	000044	$	Dollar Sign
37	022400	000045	%	Percent
38	023000	000046	&	Ampersand, And Sign
39	023400	000047	'	Apostrophe, Acute Accent
40	024000	000050	(	Left (opening) Parenthesis
41	024400	000051	)	Right (closing) Parenthesis
42	025000	000052	*	Asterisk, Star
43	025400	000053	+	Plus
44	026000	000054	,	Comma, Cedilla
45	026400	000055	–	Hyphen, Minus, Dash
46	027000	000056	.	Period, Decimal Point
47	027400	000057	/	Slash, Slant
48	030000	000060	0	
49	030400	000061	1	
50	031000	000062	2	
51	031400	000063	3	
52	032000	000064	4	Digits, Numbers
53	032400	000065	5	
54	033000	000066	6	
55	033400	000067	7	
56	034000	000070	8	
57	034400	000071	9	
58	035000	000072	:	Colon
59	035400	000073	;	Semicolon
60	036000	000074	<	Less Than
61	036400	000075	=	Equals
62	037000	000076	>	Greater Than
63	037400	000077	?	Question Mark
64	040000	000100	@	Commercial At

ASCII Character Set (continued)

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
65	040400	000101	A	
66	041000	000102	B	
67	041400	000103	C	
68	042000	000104	D	
69	042400	000105	E	
70	043000	000106	F	
71	043400	000107	G	
72	044000	000110	H	
73	044400	000111	I	
74	045000	000112	J	
75	045400	000113	K	
76	046000	000114	L	
77	046400	000115	M	Uppercase Alphabet.
78	047000	000116	N	Capital Letters
79	047400	000117	O	
80	050000	000120	P	
81	050400	000121	Q	
82	051000	000122	R	
83	051400	000123	S	
84	052000	000124	T	
85	052400	000125	U	
86	053000	000126	V	
67	053400	000127	W	
88	054000	000130	X	
89	054400	000131	Y	
90	055000	000132	Z	
91	055400	000133	[	Left (opening) Bracket
92	056000	000134	\	Backslash, Reverse Slant
93	056400	000135	]	Right (closing) Bracket
94	057000	000136	^	Caret, Circumflex
95	057400	000137	_	Underline
96	060000	000140	`	Grave Accent

ASCII Character Set (continued)

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
97	060400	000141	a	
98	061000	000142	b	
99	061400	000143	c	
100	062000	000144	d	
101	062400	000145	e	
102	063000	000146	f	
103	063400	000147	g	
104	064000	000150	h	
105	064400	000151	i	
106	065000	000152	j	
107	065400	000153	k	
108	066000	000154	l	
109	066400	000155	m	
110	067000	000156	n	Lowercase letters
111	067400	000157	o	
112	070000	000160	p	
113	070400	000161	q	
114	071000	000162	r	
115	071400	000163	s	
116	072000	000164	t	
117	072400	000165	u	
118	073000	000166	v	
119	073400	000167	w	
120	074000	000170	x	
121	074400	000171	y	
122	075000	000172	z	
123	075400	000173	{	Left (opening) Brace
124	076000	000174	:	Vertical Line
125	076400	000175	}	Right (closing) Brace
126	077000	000176	~	Tiide, Overline

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	◄──── Memory Address ────►		
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	000	†CLE	D/E	‡SL	*LS	000
			A/B	0	D/E	*RS	001		D/E		*RS	001
			A/B	0	D/E	R*L	010		D/E		R*L	010
			A/B	0	D/E	R*R	011		D/E		R*R	011
			A/B	0	D/E	*LR	100		D/E		*LR	100
			A/B	0	D/E	ER*	101		D/E		ER*	101
			A/B	0	D/E	EL*	110		D/E		EL*	110
			A/B	0	D/E	*LF	111		D/E		*LF	111
			A/B	0	D/1	L*E	101		D/1		L*E	101
			A/B	0	0/1	S*E	110		0/1		S*E	110
			NOP	000	D/E		000		D/E			000

15	14 13 12		11 10 9			8 7 6		5	4	3	2	1	0
0	ASG	000	A/B	1	CL*	01	CLE	01	SEZ	SS*	SL*	IN*	SZ*	RSS
			A/B		CM*	10	CME	10						
			A/B		CC*	11	CCE	11						

15	14 13 12		11 10 9			8 7 6		5 4 3	2 1 0
1	IOG	000		1	H/C	HLT	000		
				1	0	STF	001		
				1	1	CLF	001		
				1	0	SFC	010		
				1	0	SFS	011	◄──── Select Code ────►	
			A/B	1	H/C	MI*	100		
			A/B	1	H/C	LI*	101		
			A/B0	1	H/C	OT*	110		
			1	1	H/C	STS	111		
				1	1	CLC	111		
				1	0	STO	001	000	001
				1	1	CLO	001	000	001
				1	H/C	SOC	010	000	001
				1	H/C	SOS	011	000	001

15	14 13 12		11 10 9	8 7 6	5	4 3	2 1 0
1	EAG	000	MPY** 000	010		000	000
			DIV** 000	100		000	000
			DLD** 100	010		000	000
			DST** 100	100		000	000
			ASR 001	000			
			ASL 000	000	0	11	
			LSR 001	000	0	00	
			LSL 000	000	1	0	
			RRR 001	001	1	0	◄── Number Of Bits ──►
			RRL 000	001	0		
					0		

NOTES:	* A or B according to bit 11	†CLE	Only this bit is required
	D/I, A/B, Z/C, D/E, H/C coded 0/1	‡SL	Only this bit and bit 11
	** Second word is Memory Address		(A/B as applicable) are required

Extended Instruction Group Codes in Binary

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAX/SAY/ SBX/SBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	0	0
CAX/CAY/ CBX/CBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	0	1
LAX/LAY/LBX/ LBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	1	0
STX/STY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	0	1	1
CXA/CYA/ CXB/CYB	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	1	0	0
LDX/LDY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	1	0	1
ADX/ADY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	1	1	0
XAX/XAY XBX/XBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	1	1	1
ISX/IXY/DSC/ DSY	1	0	0	0	1	0	1	1	1	1	1	1	X/Y	0	0	I/D
JUMP INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	▨	0	1	0
BYTE INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	0	▨	▨	▨	▨
BIT INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	▨	▨	▨	▨
WORD INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	1	1	▨	▨

JUMP INSTRUCTIONS:
JLY = 0
JPY = 1

BYTE INSTRUCTIONS:
LBT = 0 1 1
SBT = 1 0 0
MBT = 1 0 1
CBT = 1 1 0
SFB = 1 1 1

BIT INSTRUCTIONS:
SBS = 0 1 1
CBS = 1 0 0
TBS = 1 0 1

WORD INSTRUCTIONS:
CMW = 0
MVW = 1

ID Segment Format for Non-CDS Programs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	List linkage														
2	TEMP 1														
3	TEMP 2														
4	TEMP 3														
5	TEMP 4														
6	TEMP 5														
7	Program Priority														
8	GOPRV nesting count														
9	Point of Suspension														
10	A–Register at Suspension														
11	B–Register at Suspension														
12	E	C	Debug Codes												0
13	Name (1st Character)								Name (2nd Character)						
14	Name (3rd Character)								Name (4th Character)						
15	Name (5th Character)								Father's ID Segment number						
16	MRD	NA	ST	BR	SC	ID	DS	OF	SS	MLD	Status				
17	Time List Link Word														
18	R	Rsltn	T	Multiple for Resolution											
19	Low order 16 bits execution time														
20	High order 16 bits execution time														
21	Timeslice Clock														
22	High main memory address +1														
23	Current overlay high address +1														
24	Number of overlays								Current overlay number						
25	AM	Size of data seg–1						High base page address							
26	Program load block number								Debug		0				
27	Program load track number														
28	Undefined							x	Disk LU for program load						
29	Sequence numb			x	IO	SV	SR		Terminal LU						
30	Temporary Data Block (TDB) List Head														
31	AD	Pointer to memory descriptor of data partition													
32	(Page offset of data partition in swap file) +1														
33	XE	EMA/WS size (including PTE)													
34	FA	PP	CB	SP	KL	FW	VM	SO	ID segment number –1						
35	NS	MSEG log page				TM	Size of MSEG			x	LE	SN	SD		
36	Highest overlay address+1 (same as 22 if unsegmented)														
37	Copy of user's WMAP														
38	Undefined														
39	DVT pointer if ID segment in use as I/O block														
40	Pointer to user ID table entry														
41	# of RNs owned and locked								# of LUs locked to program						
42	Nonbuffered I/O request count														
43	Pending I/O list head														
44	CPU usage count (high order word)														
45	CPU usage count (low order word)														

Entry point names are shown in Entry Point Names Table.
$IDA is a pointer to the first ID segment.
$ID# contains the number of ID segments.
$IDSZ contains the size of each ID segment in memory.

ID Segment Format for CDS Programs

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	List linkage														
2	TEMP 1														
3	TEMP 2														
4	TEMP 3														
5	TEMP 4														
6	TEMP 5														
7	Program Priority														
8	GOPRV nesting point														
9	Point of Suspension														
10	A-Register at Suspension														
11	B-Register at Suspension														
12	E	C	Debug Codes														0
13	Name (1st Character)									Name (2nd Character)						
14	Name (3rd Character)									Name (4th Character)						
15	Name (5th Character)									Father's ID Segment number						
16	MRD	NA	ST	BR	SC	ID	DS	OF	SS	MLD	Status					
17	Time List Link Word														
18	R	Rsltn || T	Multiple for Resolution												
19	Low order 16 bits execution time														
20	High order 16 bits execution time														
21	Timeslice Clock														
22	Link word from shared program table entry														
23	S	Shared program table pointer (0 if not shared)													
24	x	Number of code segments - 1								x	Executing seg # at susp.					
25	AM	Size of data seg - 1							High base page address						
26	Program load block number						Debug			Code block size			
27	Program load track number														
28	AL	Blocks in code partition						Disk LU for program load							
29	Sequence numb				x	IO	SV	SR	Terminal LU					
30	Temporary Data Block (TDB) List Head														
31	AD	Pointer to memory descriptor of data partition													
32	(Page offset of data partition in swap file) + 1														
33	XE	EMA/WS size (including PTE)													
34	FA	PP	CB	SP	KL	FW	VM	SO	ID segment number - 1						
35	NS	Log. start pg MSEG				TMR	Size of MSEG			X	LE	SN	SD
36	MRC	MLC	Reserved		Data seg min pos - 1			Initial code seg number					
37	Copy of user's WMAP														
38	AC	Pointer to memory descriptor of code partition													
39	DVT pointer if ID segment in use as I/O block														
40	Pointer to user ID table entry														
41	# of RNs owned and locked								# of LUs locked to program						
42	Nonbuffered I/O request count														
43	Pending I/O list head														
44	CPU usage count (high order word)														
45	CPU usage count (low order word)														

Entry point names are shown in Entry Point Names Table.

Entry Point Names

The entry point names for the non-CDS and CDS ID segments are as follows:

WORD	ENTRY POINT NAME	WORD	ENTRY POINT NAME	WORD	ENTRY POINT NAME
1	$XQT	17	$TLNK	33	$VMAS
2	$TMP1	18	$RES	34	$EMAS
3	$TMP2	19	$TIM1	35	$MSEG
4	$TMP3	20	$TIM2	36	$HSEG
5	$TMP4	21	$TICK	37	$WMAP
6	$TMP5	22	$HIGH	38	$CMD
7	$PRIO	23	$CSEG = $SHPT	39	$IOCT
8	$PENT	24	$SEGS	40	$OWNR
9	$SUSP	25	$HIBP	41	$LCNT
10	$A	26	$PART = $BLK#	42	$UIO
11	$B	27	$TRAK	43	$IO
12	$EO	28	$DISC	44	$CPUH
13	$N1.2	29	$CON	45	$CPUL
14	$N3.4	30	$TDB	46	$GPCNT
15	$N5.F	31	$MD = $DMD	47	$CPLV
16	$STAT	32	$SWP	48	$SGNL

Words Appended to ID Segment in Type 6 File

Word	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
46	Highest overlay address +1															
47	Shareable EMA reserved partition number, if assigned															
48 thru 55	Shareable EMA area label, 16 characters (zero if no shareable EMA used)															
56	Virtual memory size − 1 (in pages)															
57	x	Original CPLV						Rqus CPLV					Program CPLV				
58	Primary entry point															
59	System checksum value (same as value at $CKSM)															
60	System common checksum (same as value at $SCCK)															
61	ID checksum value (sum of Words 1 through 60)															
62	Normal primary entry point															
63	Debug primary entry point															
64	RPL checksum															
65	(reserved)															T
66	System checksum of original system															
67	$BPLO															
70	$BPHI						Transportability information									
71	$TRLO															
72	$TRHI															

Explanations for Abbreviations, ID Segment Format Tables

Word	Symb.	Explanation
12	E	E-Register at suspension.
	C	C-Register at suspension.
	O	O-Register at suspension.
16	MRD	Non-CDS: program in memory. CDS: data in memory.
	NA	No-abort; set to sign bit of EXEC request control word.
	ST	String created for program is in SAM.
	BR	Break bit; set by user break.
	SC	If set, program accesses system common.
	ID	If set, ID segment deallocated when program terminates.
	DS	If set, program has DS/1000 resources locked.
	OF	If set, system to set program to dormant state at earliest opportunity.
	SS	If set, system to suspend program at earliest opportunity.
	MLD	If set, program has executed EXEC 22 and is locked in memory.
18	R	Program time-scheduled relatively; T must be set.
	Rsltn	Resolution (00 = milliseconds; 01 = seconds; 10 = minutes; 11 = hours).
	T	Program in time list.
23	S	CDS: program is shared.
25	AM	Data segment; set if free memory in partition to be swapped with program (applies only to data partition in CDS).
26	DBG	For symbolic debug.
28	AC	CDS: all code segments are memory resident.
29	IO	Set during VMA I/O processing or non-buffered EXEC call.
	SV	Set when program terminates saving resources.
	SR	Set when program terminates serially reusable.
31	AD	If set, data is assigned to reserved partition.
32		Page offset in swap file + 1 (bits 0 - 15)
33	XE	If set, program is a Large or Extended model EMA/VMA program.

Word	Symb.	Explanation
34	FA	Set when program has made file access.
	PP	Set when program in session.
	CB	If set, program in Gocrit state.
	SP	System process if set.
	KL	If set, any user may OF program.
	FW	If set, another program waiting to terminate has scheduled program with wait.
35	NS	No suspend bit.
	TMR	Timer bit for signal processing.
36	MRC	CDS: If set, code in memory.
	MLC	CDS: If set, program has executed EXEC 22 and code segment is locked in memory.
38	AC	If set, code partition assigned to reserved partition.
47	ORG	Capability level assigned at link time.
	RQUS	Required user capability level.
	PROG	Current program capability level.
48	SD	If set, signal to be delivered.

ID Segment Extension

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	$SHEMA	SA	SAB or SHEMA table pointer, or 0 if not SHEMA													
1	$VMAS	VMA size−1, or 0 if not VMA															
2	$CPLV	x	Original CPLV				Req User CPLV					Program CPLV				
3	$SGNL	Pointer to Signal Control Block in XSAM															
4	$PENT	Primary entry point address															

Word 0:   SA =0   if bits 0-14 point to a SHEMA table entry in XSAM or are zero, indicating that this program does not use SHEMA.

   SA = 1   if bits 0-14 point to a SHEMA Association Block (SAB) in XSAM, describing multiple SHEMA associations.

   Bits 0-14   point to either a SHEMA table entry or a SHEMA Association Block in XSAM, or are zero if the program does not use SHEMA.

Word 2:   Original CPLV   = Capability assigned at link time (bits 10−14)
   Req User CPLV   = Required user capability level (bits 5−9)
   Program CPLV   = Current program capability level (bits 0−4)

Word 3:   Pointer to the program's Signal Control Block in XSAM, or zero if the program does not use signals.

Short ID Segment

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Name (1st character)								Name (2nd character)							
2	Name (3rd character)								Name (4th character)							
3	Name (5th character)								Base page block offset							
4	Overlay entry point															
5	High overlay address +1															
6	Reserved							High base page address								
7	Block offset of overlay															
8	Checksum (sum of words 1 through 7)															

$SISZ = size in words of each short ID segment

Resource Number Table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Number of Resource Numbers (n)															
RN #1	Owner ID Segment No.								Locker ID Segment No.							
RN #2	Owner ID Segment No.								Locker ID Segment No.							
	.															
	.															
	.															
RN #n	Owner ID Segment No.								Locker ID Segment No.							

$RNTA is a pointer to the first word of the Resource Number Table.

LU Table

```
┌─────────────────────────┐
│                         │
│     DVT Address LU 1     │
│                         │
├─────────────────────────┤
│                         │
│     DVT Address LU 2     │
│                         │
├─────────────────────────┤
│            .            │
│            .            │
│            .            │
│            .            │
├─────────────────────────┤
│                         │
│     DVT Address LU N     │
│                         │
└─────────────────────────┘
```

Note: Entry of zero will be interpreted
       as an unassigned device.

$LUTA is a pointer to the first word of
the LU table.

$LUT# is the number of entries in the
LUT.

Interrupt Table

Select
Code
```
     ┌─────────────────────────┐
 20  │    IFT Address or 0     │
     ├─────────────────────────┤
 21  │    IFT Address or 0     │
     ├─────────────────────────┤
  ▪  │                         │
  ▪  │                         │
  ▪  │                         │
     ├─────────────────────────┤
 77  │    IFT Address or 0     │
     └─────────────────────────┘
```

To get entry (example)

LDB SC	Get select code
ADB $INTA	Add pointer
LDA B,I	Get entry
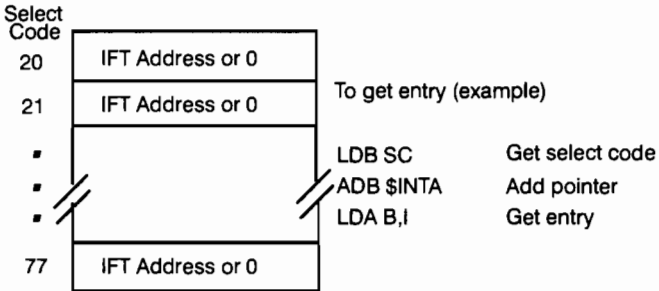
$INTA is a pointer to (first interrupt table address) − 20B
$INT# is the number of interrupt table entries.

Device Table

Row	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	DVT Link Word															
2	Q	Request Initiation List													
3	N	Circular Node List													
4	P	Circular DVT List													
5	X	Address of Interface Table													
6	AV		Device Type					Status							E
7	System Flags					LU Lock Flag							A	RS		
8	B	Buffer Limit Accumulator													
9	S	(High-Low)/16							Low Buff Limit/16						
10	x	Starting Physical Page													
11	Timeout List Linkage																
12	Device Drive Timeout Clock															0
13	Interface Driver Timeout Value																
14	Device Drive Entry Address																
15	TY		UE	Z	Subfunction						NB	X	L	UD	RQ		
16	Request Parameter #1 / Error Code with D ,F																
17	Request Parameter #2 / Transmission Log																
18	Request Parameter #3 / Extended Status #1																
19	Request Parameter #4 / Extended Status #2																
20	I	Driver Communication							Device Priority						
21	# Driver Parameters							# Extension Words								
22	DVT Extension Address																
23	Driver partition physical page																
24	M	PA	x	WA	Reserved								SLN			
25	Spool Node List																
	Start of Driver Parameter Area																
	Start of DVT Extension Area (storage)																

Words 1 through 25 are entry points $DV1 through $DV25.

$DVTP is the start of the driver parameter area.
$DVTA is a pointer to the first DVT.
$DVT# is the number of DVT entries.
$DVSZ is the size of each DVT.

Explanations for Abbreviations, Device Table (DVT)

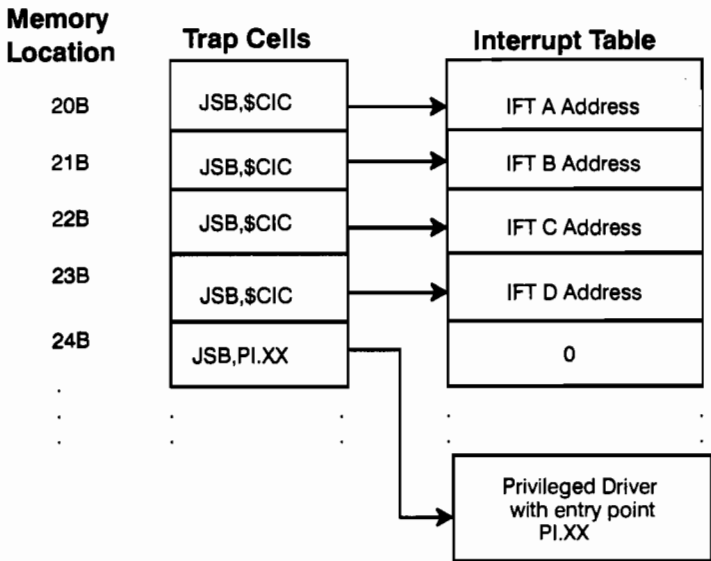| Word	Symb.	Explanation
2	Q	Queueing option, I/O requests to DVT. 0 = program priority, 1 = FIFO.
3	N	Node status. 0 = DVT not busy; 1 = busy.
4	P	Power fail flag. 0: don't call drivers on power fail; 1: call on driver for power fail operations.
6	AV	Device availability. 0 = available; 1 = down; 2 = busy; 3 = down and busy.
	E	Severe error bit.
7	A	Abort bit. 1 means system is aborting pending request.
	RS	Request status. 0 indicates request queued on IFT, request initiation list; 1 indicates request on IFT, current head of list; 2 indicates request on IFT, request complete list; 3 indicates request on DVT, request complete list.
8	B	Device buffering. 0 = unbuffered; 1 = buffered.
9	S	Buffer status. If 1, request is beyond upper limit.
15	TY	Request type. 0 = user; 1 = buffered user; 2 = system I/O; 3 = class I/O.
	UE	User error bit.
	Z	Double buffer bit.
	NB	Non-buffered bit.
	L	If set, data in user partition. If clear, data in system partition or SAM.
	UD	If set, bypass device driver and call interface driver.
	RQ	Request type. 0 = multibuffered; 1 = Read or Write/Read; 2 = Write; 3 = control.
20	I	Initial request bit. 1 indicates first time driver has serviced this device.
24	M	Location of I/O control block. 0 indicates system map; other values indicate SAM.
	PA	Pseudo abort bit. 1 = pseudo abort active.
	WA	Waiting to abort bit. 1 = abort request pending.
	SLN	System Language Number. 0 is default.

Interface Table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$IF1	Timeout List Linkage															
$IF2	Timeout Clock															
$IF3	Q	Request List Linkage														
$IF4	Interface Driver Entry Address															
$IF5	Device Table Address ($DVT1)															
$IF6	AV		Interface Type						WA	X	I/O Select Code					
$IF7	System Flags					F	M	# Words IFT Extension								
$IF8	Driver Partition Physical Page															
$IF9	MA	X	ML	OH	MQ	Reserved							Map Set #			
$IFTX	Start of IFT Extension (Storage)															

$IFTA is a pointer to the first IFT
$IFT# is the number of IFTs
$IFSZ is the size of each IFT

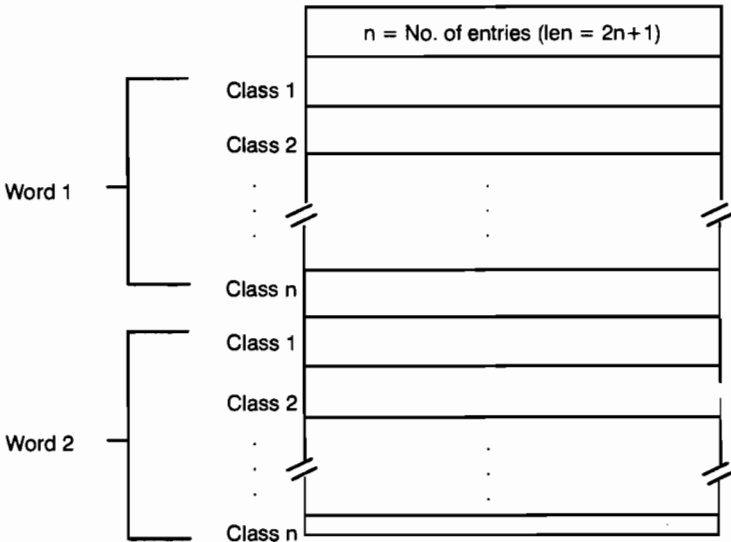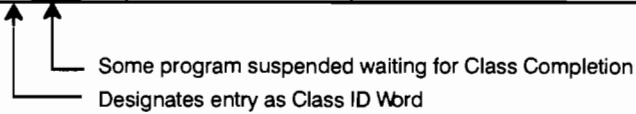| Word	Symb.	Explanation
3	Q	Queueing option of DVTs with active requests. 0 = device priority; 1 = FIFO.
6	AV	Interface availability. 0 = available; 1 = locked to DVT; 2 = busy; 3 = locked and busy.
	WA	Waiting-to-abort bit. If 1, abort request pending.
7	F	First entry bit. 1 indicates first time driver has serviced request on this IFT.
	M	If 1, interface driver has dequeued list.
9	MA	Map allocated bit. If 1, map set allocated for this I/O channel.
	ML	Map locked bit. If 0, map is dynamically allocated and deallocated. If 1, $MSRTN does not deallocate map set.
	OH	On hold bit. Has copy of H bit from system flags in $IFT7.
	MQ	Map set queued bit. If 1, request is in map set suspend queue waiting until map set is available.

Trap Cells and the Interrupt Table

Memory Location	Trap Cells	Interrupt Table
20B	JSB,$CIC	→ IFT A Address
21B	JSB,$CIC	→ IFT B Address
22B	JSB,$CIC	→ IFT C Address
23B	JSB,$CIC	→ IFT D Address
24B	JSB,PI.XX	0

Privileged Driver with entry point PI.XX

Class Table

n = No. of entries (len = 2n+1)

Word 1
- Class 1
- Class 2
- .
- .
- .
- Class n

Word 2
- Class 1
- Class 2
- .
- .
- .
- Class n

$CLTA is a pointer to the first word of the Class Table.

Class ID Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	W	Res	Low 5 bits of ID Segment Number					No. Pending Class Requests (0−255)							

Some program suspended waiting for Class Completion
Designates entry as Class ID Word

Swap Descriptor Table

Word 0	Link Word
Word 1	Page Offset into Swap Area
Word 2	Size of Free Swap Area in Pages

$SWPS: Pointer to free swap area descriptor prior to the one at which to start a search for a free area.
$FSWP: Head of the free swap area descriptor list.
$UFSD: Head of the unused free swap area descriptor list.
$SWLU: LU of swapping disk.
BLK/T: Number of 128-word block per track.

Shareable EMA Table Entry

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	XSAM address of the next SHEMA table entry, or zero															
1	LK	IX	IN	IL	reserved					SHEMA # for LEMA						XX
2	XSAM address of the previous SHEMA table entry, or zero															
3	Allocation															
4-11	SHEMA label (16-characters, blank-extended)															
12	# in system (# IDs)								In-use count (# active)							

Word 1: LK = 1   if SHEMA partition is locked

IX = 1   if SHEMA partition has been initialized as an Extended SHEMA, or as a Large SHEMA if IL = 1

IN = 1   if SHEMA partition has been initialized as a Normal SHEMA

IL = 1   if SHEMA partition has been initialized as a Large SHEMA, in which case IX will also be = 1

XX = 1   if an extra word of XSAM was allocated for this SHEMA table entry (that is, 14 words were allocated)

For Large EMA programs only, "SHEMA # for LEMA" is set to the EMA segment number minus one for which this SHEMA has been initialized.

Word 3:   Specifies the allocation of the area:

0 =   Shareable EMA partition is not allocated; it is to be allocated from dynamic memory

1-1023 =   Shareable EMA partition is not allocated; it is to be allocated in this reserved partition number

1024 and
above =   Shareable EMA partition is allocated; this word holds the address of the Memory Descriptor (MD) for the partition

Word 12:   "# in system" count tells how many programs are RP'd that use the SHEMA.

"in-use count" tells how many of the RP'ed programs are active.

SHEMA Association Block (SAB)

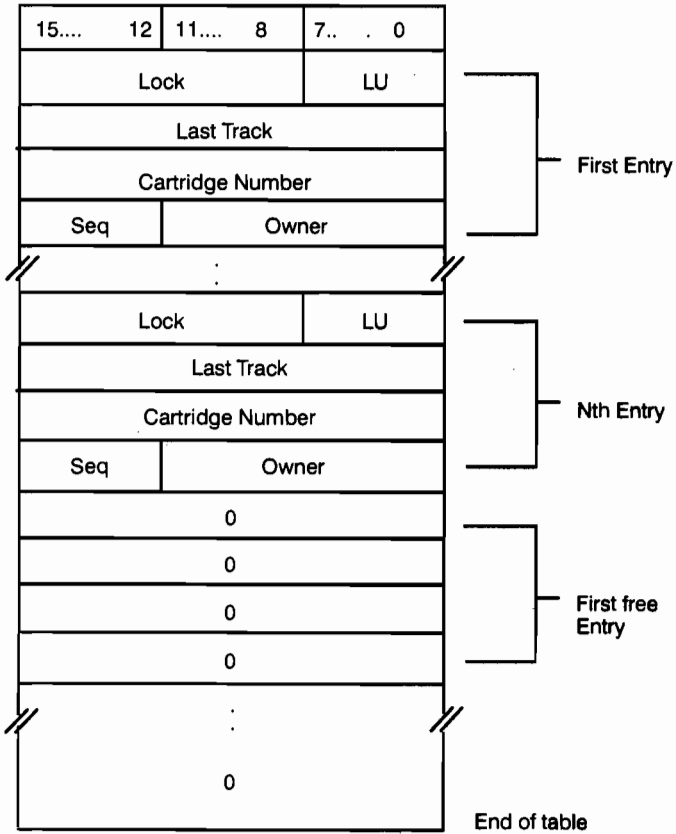|	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SX	pointer to next SAB, or 0 for end of list														
1	pointer to SHEMA table entry, or 0 if unused															
2	pointer to SHEMA table entry, or 0 if unused															
· · ·	· · ·															
8	pointer to SHEMA table entry, or 0 if unused															

Word 0:   SX: 1 if an extra word was allocated in XSAM for the SAB

Bits 0-14:   Contain XSAM address of next SAB in list for associated program, or zero if this is the last SAB for the program.

Words 1-8:   Each word contains the XSAM address of a SHEMA Table Entry that the associated program has attached to itself, or zero if the SAB entry is unused.

I/O Control Block

TY=0 Normal Request Stored in ID Segment starting at $XQT	Word	TY=2 System Request Stored in XSIO Block
	−1	LU
	0	Completion Address
I/O List Linkage	1	I/O List Linkage
I/O Control Word	2	I/O Control Word
PRAM1 / Buffer Address	3	PRAM1 / Buffer Address
PRAM2 / Buffer Length	4	PRAM2 / Buffer Length
PRAM3 / Z-Buffer Address	5	PRAM3 / Z-Buffer Address
PRAM4 / Z-Buffer Length	6	PRAM4 / Z-Buffer Length
Priority	7	Priority
	8	Starting Phys. Page of Data
	9	Status Return
	10	Transmission Log

Buffered Request TY=1 Stored in SAM	Word	Class Request TY=3 Stored in SAM
I/O List Linkage	1	I/O List Linkage
I/O Control Word	2	I/O Control Word
PRAM1 / Buffer Address	3	PRAM1 / Buffer Address
PRAM2 / Buffer Length	4	PRAM2 / Buffer Length
PRAM3 / Z-Buffer Address	5	PRAM3 / Z-Buffer Address
PRAM4 / Z-Buffer Length	6	PRAM4 / Z-Buffer Length
Priority	7	Priority
I/O Block Length	8	I/O Block Length
ID Segment and Run Numbers	9	Class Information
Undefined	10	User Defined Value
Undefined	11	ID Segment Address
Undefined	12	VMAIO Control Word
Undefined	13	VMAIO Z-Buffer Address
ID Segment Forward Pointer	14	ID Segment Forward Pointer
ID Segment Backward Pointer	15	ID Segment Backward Pointer
DVT Address	16	DVT Address/Previous Length
Data	17	Data
.	.	.
.	.	.
.	.	.
Last Word of Data		Last Word of Data
	Block Length	

I/O Control Word	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TY		UE	Z			Subfunction				NB	0			UD	RQ

Cartridge Directory

15.... 12	11.... 8	7.. . 0	
Lock		LU	First Entry
Last Track			
Cartridge Number			
Seq	Owner		
:			
Lock		LU	Nth Entry
Last Track			
Cartridge Number			
Seq	Owner		
0			First free Entry
0			
0			
0			
:			
0			End of table

$CDA is a pointer to the start of the directory.
$CD# is the number of entries.
$CDSZ is the size of each cartridge directory entry.

Dynamic Memory Descriptor

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Word 0	Number of Pages in Block
Word 1	Starting Page of Block
Word 2	0 | ID Segment Address of Occupant or 0 | 00000B = In use as Shareable EMA Partition or 1 | 77777B = Partition Bad (Unusable) or 1 | Pointer to Previous Free Block
Word 3	C/D | reserved | IO | LK | OV
Word 4	Pointer to Next Free Block or Priority of Occupant
Word 5	Pointer to Previous Adjacent Block
Word 6	Pointer to Next Adjacent Block

$STMD Start Search pointer to dynamic memory descriptors.

Reserved Partition Memory Descriptor

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Word 0	Number of Pages in Block
Word 1	Starting Page of Block
Word 2	0 | ID Segment Address of Occupant or 0 | 00000B = In use as Shareable EMA Partition or 1 | 00000B = Partition Free or 1 | 77777B = Partition Bad (Unusable)
Word 3	C/D | Reserved | IO | LK | OV

$RPT# – Contains the number of reserved memory descriptors.

User ID Entry

Words 1–8	User Logon Name (16 Characters Max)

Word 9	S	Session Sequence Number	Status

S = Superuser Bit.

Words 10–11	Pointer to Working Directory or −1 If None

Word 12	Terminal LU of the User or Session #

Word 13	L	Number of User Programs Counter

L = Logoff Program/Command File Bit.

Word 14	User Identification Number

Words 15–16	Saved Password for the LOGON Program or Logon Time in seconds Since Jan 1, 1970 for Logged on User

Words 17–18	Session CPU Usage in Tens of Milliseconds

Word 19	Address of UDSP Table in XSAM

Word 20	ID Segment Address of the First Session Program

Word 21	Group ID Number	User Capability Level

Word 22	Reserved

$UIDA is a pointer to the user ID Table.
$UID# is the number of entries.
$USZ is the size of each user ID entry.

Group Configuration File

Block 1

Word	Content
1	Group Identification Number
2-3	Cumulative CPU Usage in Tens of Milliseconds
4-5	Cumulative Connect Time in Seconds
6-7	CPU Usage Limit in Tens of Milliseconds
8-9	Connect Time Limit in Seconds
10-25	LU Bit Map
26	Block number of the first block in the group members list
27	Number of entries in the list of members
28 : 127	Reserved (set to zero)
128	Checksum of all previous words

Block 2-N

Word	Content
1-8	User Logon Name of Member 1 (16 Chars Max) −or− Word one equals 0; signals an empty record
9	Index into Member 1's User Configuration File group list
10−17	User Logon Name of Member 2 (16 Chars Max) −or− Word one equals 0; signals an empty record
18	Index into Member 2's User Configuration File group list
:	
127-128	Reserved (set to zero)

User Configuration File

Block 1

Word	Content	
1–15	User Name (30 characters max)	
16	S	Reserved
17-18	Encoded Password (14 chars max) or a constant	
19	User Identification Number	
20	UDSP depth	Number of UDSPs
21–60	Program to run at logon (80 chars max)	
61–92	Working Directory name (64 chars max) or −1	
93–94	Last Logoff time, in seconds since Jan 1, 1970	
95–96	Cumulative connect time in seconds	
97–98	Cumulative CPU usage in tens of milliseconds	
99	Reserved	User Capability Level
100–115	LU Bit Map	
116	Block number of the first block in the users group list	
117	Number of entries (records) in the group list	
118	Index from the base of the group list to the record of the default logon group	
119–120	Last Logon Time, in seconds since Jan 1, 1970	
121	Group ID number last logged on with	
122	LU last logged onto	
123		EVB size in pages
124 to 126	Reserved (set to zero)	
127	Code word to signal if converted yet	
128	Check sum of all previous words	

Blocks 2-N

Word	Content
1–8	Group Name (16 characters max) −or− Word one equals 0; signals empty block
9–10	Cumulative CPU Usage in tens of milliseconds for the user in this group
11–12	Cumulative Connect Time in seconds for the user in this group
13–14	CPU Usage limit for the user in this group
15–16	Connect Time limit for the user in this group
17–56	Program to run at Logon (80 characters max)
57–96	Program/command−file to run at logoff (80 characters max)
97–128	Working directory name (64 chars max) or −1

MasterAccount File

Record 1	Word 1 Last Assigned User ID Number
Record 2	Name=system; User ID #2; Reserved for System
Record 3	Logon Name:User ID #3: System Manager Only
Record 4	Logon Name Corresponding to User ID #4
Record 5	Logon Name Corresponding to User ID #5
⋮	⋮
Record n	Logon Name Corresponding to User ID #n

MasterGroup File

Record	Content
1	Word 1 contains the last assigned group ID
2	Logon name corresponding to group ID 2; Reserved for system
⋮	
n	Logon name corresponding to group ID n

Code Segment Table (SRT Table)

Entry for
Code Seg-
ment 0
(four words)

15 db	14 ab	Reserved	Up to 128 CST Entries 8 ——————————— 0 Segment Length (in 128-word blocks)	Word 3
Start Track of Segment				Word 2
Disk LU		Start Block of Segment		Word 1
PA	ML	Start Physical Page Number of Segment		Word 0

where: db,ab   =   Bits 15 and 14: used by the
                              Symbolic Debug/1000.

          PA   =   Bit 15: 0 if segment is in memory,
                              1 if segment is absent.

          ML   =   Bit 14: 1 if segment is locked (by
                              the LINK ML command).

Segment Replacement Table

Block n : : : :	
Segment # of Current Code Segment in Block 2	SRT Entry 2
Segment # of Current Code Segment in Block 1	SRT Entry 1
Segment # of Current Code Segment in Block 0	SRT Entry 0
Segment Replace Table Pointer	1000B

Shared Program Table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Word 0	Disk LU								Block #								One
Word 1	Track Number																Entry
Word 2	In-System Count								In-Use Count								per Shared
Word 3	MD Pointer to Code Partition																Program
Word 4	Shared Program List Pointer																

$SPTB is the address of the first shared program table entry
$SPR# is the number of shared programs.

Disk Volume Header

Word: 1   8
| Uppercase ASCII "VOLUME HEADER   " (no parity) |

Word: 9   10   11   12   16
| ^ Bit Map | B/b | Res | Unused |

Word: 17   24
| Unused |

Word: 25   32
| Unused |

Word: 33   35   40
| Unused | ^ root dir | Unused |

Word: 41   48
| Unused |

Word: 49   56
| Unused |

Word: 57   64
| Unused |

Directory Structure

Entry

		Block Boundary
1	Directory Header	
2	File or Extent Entry	
3	File or Extent Entry	
4	File or Extent Entry	
	$\vdots$	
$4n-3$	File or Extent Entry	
$4n-2$	File or Extent Entry	
$4n-1$	File or Extent Entry	
$4n$	Directory Trailer	

		Block Boundary
$4n+1$	Directory Header	
$4n+2$	File or Extent Entry	
$4n+3$	File or Extent Entry	
$4n+4$	File or Extent Entry	
	$\vdots$	EXTENT
$4(n+x)-3$	File or Extent Entry	
$4(n+x)-2$	File or Extent Entry	
$4(n+x)-1$	File or Extent Entry	
$4(n+x)$	Directory Trailer	

Root Directory Header/Trailer

Word:	1	2	3	4	5	6	7	8
	Flag	Size	Tag		^ Other		^ Volhead	

Word:	9							16
	Unused							

Word:	17	18						24
	Owner	Group	Unused					

Word:	25							32
	Unused							

Root Directory Entry

Word:	1	2	3	4	5	6	7	8
	Flag	2	^ Directory		Size	32	−1	−1

Word:	9							16
	16-Character Name							

Word:	17	18	19	20	21	22	23	24
	"DIR "		Create Time		Access Time		Update Time	

Word:	25							32
	Unused							

Directory Header/Trailer

Word:	1	2	3	4	5	6	7	8
	Flag	Size	Tag		^ Other		^ Parent	

Word:	9							16
	16-Character Name							

Word:	17	18	19	20	21	22	23	24
	Owner	Group	Unused					

Word:	25							32
	Unused							

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	

Subdirectory Entry

Word:	1	2	3	4	5	6	7	8
	Flag	2	^ Directory		Size	32	−1	−1

Word:	9							16
	16-Character Name							

Word:	17	18	19	20	21	22	23	24
	"DIR"		Time Stamps					

Word:	25	26	27	28	29	30	31	32
	Unused							

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	

Disk File DCB

Word	
0	Directory Offset/Sector/LU
1	Directory Track
2	File Type
3	Extent Base Track
4	Extent Base Sector
5	File Size
6	Type 2 Record Length
7	DCB Size and Flags
8	Sectors per Track
9	Program ID Segment Address
10	Current Position Track
11	Current Position Sector
12	Index of Current Word
13 / 14	32-Bit Record Number
15	Extent Number

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

FMGR Cartridge Header

word		field
0	1	name
1		
2		
3		crn
4		frstr
5	R	nexsc
6	R	sc/tr
7		lastr
8		n# trk
9		nextr
10		badtr
11		
12		
13		
14		
15		

name: cartridge label (6 chars)

crn: Cartridge Reference Number

frstr: First available FMP track

R: Reserved

nexsc: next available sector

sc/tr: Sectors per Track

lastr: Last Available FMP track (+1)

n# trk: Negative Number of Directory Tracks

nextr: Next Available Track

badtr: Bad Track List (6 words)

FMGR Disk File Entry

0	1 name
1	
2	
3	type
4	track
5	ext / sectr
6	size
7	recln
8	seccd
9	flags
10	
11	
12	
13	
14	
15	

name: file name (6 chars)

type: File Type (int)

track: Starting Track

ext: Extent Number (main = 0)

sectr: Starting Sector

size: Extent Size (+sectors or
        -sector*256)

recln: Record Length (in words)

seccd: Security Code

flags: Open Flags (7 words)

FMGR File Extent Entry

0	0 name
1	
2	
3	X
4	track
5	ext sectr
6	size
7	X
8	
9	
10	
11	
12	
13	
14	
15	

name:	file name (6 chars)
X:	Not used
track:	Starting Track
ext:	Extent Number
sectr:	Starting Sector
size:	Extent Size (in sectors)
X:	Not used

Nondisk File Entry

0	0	name
1		
2		
3	0	
4	acscd	LU
5	eofcd	LU
6	spcod	
7	rwcod	
8	seccd	
9	flags	
10		
11		
12		
13		
14		
15		

name:	file name (6 chars)
acscd:	Device Access Subfunction
LU:	Device Logical Unit Number
eofcd:	EOF Subfunction
spcod:	Spacing Subfunction
rwcod:	Read/Write Subfunction
seccd:	Security Code (int or 2 chars)
flags:	Open Flags (7 words)

13

Error Messages

Build Errors

Bad Parameter Type

Bad parameter type in the runstring

Numeric runstring parameter where the file namr (or null
parameter) was expected, or lack of fixed position instruction.

Cannot RP a program with overlays

Corrupted program file

Incorrect checksum or corrupted file.

Default error path (abort) used

The error path specified in the runstring was not /C, /E, or /A.

Illegal snapshot

Incorrect snapshot header checksum, or label required by BUILD
was not found.

Incorrect file type

BUILD will only RP a type 6 file.

Invalid command in present context

The command was not recognized by BUILD.

No free ID segment

No free partition large enough for program

No free partition

All partitions have been used. If auto partitioning, there are no
more memory descriptors.

No memory descriptor available for shareable EMA area

RPd program accesses shareable EMA. No more memory
descriptors.

Not a system image

Not enough memory for program

Not enough remaining memory for shareable EMA area

Output file too small for system

Parameter out of range

Partition in use

Partition is too small

Partition number specified with PA command is too small for
program.

Build Errors (continued)

Program already loaded

Program not transportable, must be reloaded

Can result from using system common. Program may access system entry points which are not transportable.

Ran out of disk space <filename>

Shared program table full, program not shared

Specified size is smaller than the minimum required

System not for snap

The system image was not for the snapshot file given.

System not generated for CDS programs

Program uses CDS instructions, but system lacks the necessary routines to run CDS programs.

*warning – RPL checksum does not match

Program was loaded but routines that should have been RPLd may not have been replaced properly.

CI Error Messages

CI error messages are self-explanatory. For further information, refer to Appendix A of the *RTE-A User's Manual*.

COPYL Error Messages

BAD PARAMETER FORMAT.
FORMAT IS FROM,TO ie 32,31

Interactive input was not properly entered.

CONTENTS OF LU xx WILL BE DESTROYED.
ENTER "GO" TO PROCEED >

The contents of the destination LU will be lost and replaced by the contents of the source LU. Entering GO will begin the copy.

COPYL EXITING; VERIFY THAT "SYSTEM" DISK IS IN PLACE
ENTER "GO" WHEN READY.

The COPYL program is exiting.

DISK IS NOT UP AND AVAILABLE

One of the disk LUs is down or busy with another program.

DISK SIZES DO NOT MATCH

The number of tracks and the number of sectors/track must be the same for both disks.

ENTER "FROM" LU, "TO" LU (/E TO EXIT) >

This is the interactive prompt from COPYL for source and destination LUs.

INSUFFICIENT MEMORY IN PARTITION.
RESIZE PROGRAM AND TRY AGAIN.

There is not enough memory behind the program for a minimum buffer (128 words).

NOT A DISK LU

The device type recorded in the DVT indicates that the source or destination device is not a disk or the source and destination are not the same device type.

"TO" DISK MUST BE DISMOUNTED.

The "to" (destination) disk was not dismounted before running the program.

Device Driver Error Codes

The following I/O errors can occur when a device driver detects an error condition.

The error message format is:

I/O Device Error on LU XX ErrorDescription

XX = Logical unit number (LU)

ErrorDescription = One of the following:

I/O request error
Device not ready
Device timed out
End of tape detected
Transmission error
Device is write protected
Addressing error
Serial poll error
Group poll error
Driver fault
Data communication error
Device information specified at generation time is wrong

Special Driver Defined Error = nn

System does not recognize the error number, nn, reported by the driver.

Request has been flushed

Reported in addition to the above messages when the offending I/O request will not be retired.

EDIT/1000 Error Messages

This section provides brief descriptions of some of the more common error and information messages. Refer to the EDIT/1000 for corrective action.

/FX
?^

> If EDIT finds a syntax error, it reprints the command, followed by a line with a question mark and a caret at the incorrect item.

<aaaa> line not found
O saves original text written to screen
S saves text just read from screen
B saves both (inserts screen text before original text)
What should be saved?

> Missing line in screen mode. This message displayed when attempted to read screen with either a start or stop line not present. <aaaaa> is "Start" and/or "Stop".

Abort 000xx

> Internal or corrupt scratch file error; displayed when EDIT is aborted.

> xx   Error number; see ?AB for explanation of numbers.

No such file xxxxx
An ER or the first WR will create it.

> EDIT was invoked with the name of a nonexistent file in the runstring, or FI specified the name of a nonexistent file.

> xxxxx = filename specified.

Auto LF must be off

> Startup error message displayed when EDIT is run on a terminal that has the automatic line feed enabled. Turn off LF.

Block mode must be off

> Startup message when EDIT run on a terminal that has block mode enabled. Turn off block mode.

Break – Workspace unchanged.

> Informational message displayed when user cancels a command prior to execution using the break.

Cannot recover.

> Error message displayed during recovery operation indicating a severely corrupted file. To avoid reentering recover mode, purge file.

EDIT/1000 Error Messages (continued)

Command not executed.

Informational message displayed after any response but Y (YE, YES) to the OK? prompt.

Device is locked. Is waiting OK?

Displayed when listing to a device other than the terminal or a file.

EDIT aborted by user

Displayed in response to the A (Abort) command.

Entering recover mode.

Informational message indicating that the last termination of EDIT was abnormal and EDIT is entering recovery mode to re-construct the work file that was current when the abnormal termination occurred.

FMP errors: EDIT displays a one-line message for common errors, and for uncommon errors it displays:

FMP error −XXXXXX, FILENAME:SC:CR

FNAME	name of file for which error occurred.
SC	Security code.
CR	CRN.

Common FMP error messages:

File already exists FILENAME:SC:CR
No such file FILENAME:SC:CR
Illegal access FILENAME:SC:CR
File already open FILENAME:SC:CR
File already closed FILENAME:SC:CR
Directory full, corrupted file FILENAME:SC:CR
Illegal file name FILENAME:SC:CR
No such cartridge FILENAME:SC:CR
Disk full, corrupted file FILENAME:SC:CR

File is write protected

EDIT read in a write-protected file that was entered without a security code or with a wrong code.

Line limit exceeded.

An EDIT file cannot exceed 32500 lines/records.

Line nnnnnn patched

Displayed when the recover operation may have altered a line. The line number is provided. Check sequence and content of lines around the one specified.

EDIT/1000 Error Messages (continued)

Line order patched at line nnnnnn

Displayed when the recover operation may have disturbed line order. Line number provided. Check sequence and content of lines around the one specified.

nnnnnn lines recovered.

Informational message printed following the recovery operation to indicate the number of lines recovered in the work file.

No changes allowed during recover mode.

Displayed when an attempt was made to edit the file or enter screen mode during a recovery operation.

No help for XX. Use ? for general help.

No help explanation exists for XX (the first two letters of the word for which help was requested).

No marks

Displayed in response to SM MA (show marks) command when no marked lines are in file.

non HP26XX terminal – screen mode unavailable.

EDIT was called up from a terminal in which screen mode cannot be used.

No space in scratch file.

Issued in response to an attempt to modify work file after the "Out of disk work space" message is displayed.

Not an option. Type SH to show all options and their current values.

Not found.

The pattern in a forward (") or backward(") line specification search was not found.

OK?

Preceding command deletes or modifies data. Enter Y[ES] to continue, N[O] (or RETURN) to abort the command.

Operator break.

Displayed when user interrupts an EDIT operation (with a break mode BR command).

Out of disk work space.

No additional disk space available in the directory that contains the work file.

EDIT/1000 Error Messages (continued)

Read terminated before end of file found.

EDIT could not read all of a possibly corrupt file.

Setting terminal straps to d g l T

Startup message when EDIT is run on a terminal not set up correctly. Indicates that EDIT is adjusting the terminal programmatically.

Size Edit up.

EDIT was not loaded properly. SZ command was not used when EDIT was loaded to set proper size.

Sorry, no help. File "EDIT. not found.

EDIT was not installed properly. Unless the file "EDIT. is in the system no help messages can be given for help commands (?, H, HE).

Start > EOF

An edit operation was attempted over a line range specifying a start line beyond the end of the file.

Start > stop

An edit operation was attempted over a line range specifying a start line beyond the last line specified.

Work file FILENAME

Informational message displayed during recovery operation describing the location and name of the EDIT work file.

Work file error. ?

Work file,FILENAME:

Internal error. Workfile is probably corrupt.

EXEC Error Codes

EXEC errors cause the calling program to abort and the following message to appear on the terminal screen, unless the no-abort option was used in the EXEC call.

name ABORTED CLxx address [segment]
name ABORTED IOxx address [segment]
name ABORTED LDxx address [segment]
name ABORTED LUxx address [segment]
name ABORTED MP pCounter [segment]
name ABORTED OPxx address [segment]
name ABORTED RNxx address [segment]
name ABORTED RQ address [segment]
name ABORTED SCxx address [segment]
name ABORTED SGxx address [segment]
name ABORTED SR pCounter [segment]
name ABORTED SW address [segment]
name ABORTED UI address

name	Name of the program.
address	Memory address of JSB EXEC that caused the violation.
pCounter	Current program counter value.
segment	Number of the segment most recently loaded (0 for nonsegmented programs).
xx	Two-digit error code number.

CLRQ Error Codes

CL01	Illegal class number or no class table.
CL02	Parameter or calling sequence error.

CS Error Codes

CS00	CDS software not installed.
CS01	Segment load requested by instruction in data segment.
CS02	CST index out of bounds.
CS03	Invalid SST entry.
CS04	Too many interrupts.
CS05	CDS program is corrupt, internal error found.
CS06	Stack overflow. LINK program with more stack space.

EXEC Error Codes (continued)

EV Error Codes

EV00	Invalid environment variable block.
EV01	Incorrect number of parameters.
EV02	Illegal parameter value.
EV04	Environment variable block is busy (no-suspend bit set).

IO Error Codes

IO00	Illegal class number.
IO01	Not enough parameters in I/O call, or illegal access to the disk, or illegal disk subfunction specified for a nondisk device.
IO02	Illegal LU in I/O call, or illegal class request to disk.
IO04	Illegal buffer address in I/O call or not enough SAM (now or ever).
IO07	Driver has rejected the call.
IO10	Illegal Class Get.
IO11	Attempt to input to spooled LU.
IO13	LU is locked.
IO14	LU is down.

LD Error Codes

LD	(Load). The program load (possibly a segment load or swap in) failed.

LU Error Codes

LU02	Illegal LU: greater than maximum, less than 1, or not assigned.

MP Error Codes

MP	(Memory Protect). A memory protect is illegal; the program is aborted.

EXEC Error Codes (continued)

Option Error Codes

A request to an optional EXEC call/OS command that was not included in the system.

OP00	Module for this OS command not present
OP08	No EXEC 8 capability (Segment load)
OP09	No EXEC 9 capability (Schedule with wait)
OP10	No EXEC 10 capability (Schedule no wait)
OP12	No EXEC 12 capability (Time schedule)
OP17	No EXEC 17 capability (Class read)
OP18	No EXEC 18 capability (Class write)
OP19	No EXEC 19 capability (Class control)
OP20	No EXEC 20 capability (Class write/read)
OP21	No EXEC 21 capability (Class Get)
OP23	No EXEC 23 capability (Queue schedule with wait)
OP24	No EXEC 24 capability (Queue schedule no wait)
OP39	%ENVRN is not generated into the system.

RN Error Codes

RN00	No option bits set in call; when requesting a resource number, the control word must have one of bits 0 through 5 set.
RN01	Invalid number of parameters.
RN02	Undefined resource number.
RN03	Attempt to clear on RN not locked to this program, or invalid RN.

RQ Error Codes

RQ	(Request). An EXEC request with illegal request code or with more than eight parameters.

EXEC Error Codes (continued)

SC Error Codes

SC   (Scheduling). A scheduling error, according to the following table:

		EXEC CALLS
SC01	Not enough parameters	11, 14
SC02	Illegal parameter value	12, 14
SC03	SECURITY VIOLATION detected	
	Insufficient capability to schedule program	9, 10, 12 23, 24
	−or−	
	Attempted to access an LU that is not in the session LU access table	1, 2, 3, 17, 18, 19, 20
SC04	Illegal buffer address or length (outside the user partition or partially in system common or program to terminate is not a son)	8, 9, 10, 11, 12, 14, 23, 24, 28, 6
SC05	Program or segment name not found	8, 9, 10, 12, 23, 24, 28
SC09	Program is too large to fit in largest usable block of dynamic memory	
SC10	Never enough memory (SAM) to pass the desired string	9, 10, 14, 23, 24
SC15	Not enough memory (SAM) to pass string (no-suspend bit set)	9, 10, 14, 22, 23, 24

EXEC Error Codes (continued)

SG Error Codes

SG01	Illegal number of parameters on EXEC call.
SG02	Illegal parameter on an EXEC call.
SG03	Cannot use SglPause while in handler.
SG04	Not enough XSAM at this time (usually when calling SglHandler).
SG05	Not enough capability to deliver signal when using SglKill.
SG06	Buffer limit suspended when using SglKill.
SG07	Unexpected signal received.
SG08	No such program as identified by call to SglKill.
SG09	Illegal buffer descriptors when operating system attempted to deliver signal (cannot be trapped with noabort).

SR Error Codes

SR	(Subroutine). A program attempted to call a routine on a higher numbered level than the currently executing routine; the program is aborted.

SW Error Codes

SW	(Swap). The program swap out failed.

UI Error Codes

UI	(Unimplemented Instruction). Execution, in a user program, of an instruction the computer does not recognize.

FC Errors

attempting to use data in spite of checksum error

See "checksum error (chunk body)."

bad chunk header is:
< chunk size > < chunk type > < #entries > < 1st file # or 0 >

Same as "bad tape cartridge entry."

bad microdirectory entry is:
< file# > < file blk > < #blks > < buffer block >

Same as "bad tape cartridge entry."

bad record length

Record length indicated by the transmission log returned from the driver disagreed with the length value specified in the header field of the record.

bad tape cartridge entry is:
< octal dump >

This message may accompany a tape format error message and is for diagnostic use only.

bad tape diskfile entry is:
< octal dump >

Same as "bad tape cartridge entry."

can't load segment < segment name >

can't open < name >::< CRN > due to error:
FMGR< nnn >

The indicated source file could not be opened to copy.

cartridge clearing error on disk LU nnn:
FMGR < nnn >

Same as "cartridge lock error...", but for clearing the directory on the disk cartridge.

cartridge lock error on disk LU nnn:
FMGR < nnn >

FC was unable to lock the indicated disk cartridge due to FMGR error.

cartridge unlock error on disk LU nnn:
FMGR < nnn >

Same as "cartridge lock error...", but for cartridge unlock.

checksum error (chunk body)

Same as "checksum error (chunk header)" unless the I option was specified, in which case, FC may attempt to use the information.

FC Errors (continued)

checksum error (chunk header)

FC's checksum information indicated a problem; if the retries fail, FC considers the information to be lost.

command file error:
FMGR <nnn>

FMGR error nnn occurred when FC attempted to read a record from the command file.

comment file error:
FMGR <nnn>

FMGR error nnn occurred when the comment file was opened or read.

copy terminated

A copy operation terminated before completion.

CRN <crn> LU nnn not large enough:
data blocks/dir entries needed:<bl>/<ent>,available: <bl>/<ent>

The specified cartridge does not have file or directory space for the files to be copied from the tape.

CRN <crn> not found

A tape-to-disk COPY command with an unspecified destination cartridge was issued, and the required CRN was not found.

data lost

Similar to "file too large..." error, except that the "data lost" error occurred while reading the source tape rather than in a FMGR routine.

<source namr> [<dest namr>]
data lost due to disk error when tape was made

Same as "data lost" except the data was lost because of a disk error that occurred when the tape was written.

<source namr> [<dest namr>]
data lost due to disk error when tape was made:
[xtnt <x1> blk <b1> thru [xtnt <xn>] blk <bn>

Same as "data lost due to disk error," except that only part of the file was lost.

data lost due to disk error when tape was made: entire file

Same as "data lost due to disk error when tape was made," except the I option was used to attempt to recover part of the file, but the attempt did not succeed.

FC Errors (continued)

<source namr>[<dest namr>]
data lost for file, reference number = nnn

Data was lost for file reference number nnn.

data lost:
[xtnt <x1>] blk <b1> thru [xtnt <xn>] blk <bn>

Same as "data lost: entire file," except that only part of the file was lost.

<source namr>[<dest namr>]
data lost: entire file

Same as "data lost," except the I option was used to attempt to recover part of the file, but the attempt did not succeed.

<source namr>[<dest namr>]
disk directory read failed: LU nnn dir tr nnn [thru dir tr nnn] error reading
directory on disk LU nnn:
FMGR <nnn>

Same as "...write...", but for directory read instead of write.

disk directory read required retries:
LU nnn dir tr nnn thru dir tr nnn

Same as "disk write required retries," but for disk read in the directory area.

disk directory write failed: LU nnn dir tr nnn [thru dir tr nnn] error writing
directory on disk LU nnn:
FMGR <nnn>

A directory write to the specified track or tracks failed due to a disk error (FMGR-001) or verify error (FMGR-049), even after retries were attempted.

disk directory write required retries:
LU nnn dir tr nnn thru dir tr nnn

Same as "disk write required retries" but for disk write in the directory area.

disk read failed: LU nnn trk nnn sec nnn thru trk nnn sec
nnn

A disk read failed, even after retries.

Disk read required retries:
LU nnn trk nnn sec nnn thru trk nnn sec nnn

Same as "disk write require retries" but for disk read.

Disk write failed:
LU nnn trk nnn sec nnn thru trk nnn sec nnn

A disk write failed, even after retries were done. FC also reports a disk error (FMGR-001) for each file affected.

FC Errors (continued)

disk write required retries:
LU nnn trk nnn sec nnn thru trk nnn sec nnn

> The indicated disk write was unsuccessful on the first try, but succeeded after it was retried one or more times.

DO YOU REALLY WANT TO PURGE DISK LU <#>, CRN <crn>?

> A COpy command specified the C option to clear the destination cartridge of all current data.

error: already in group

> A GROUP command was entered before the previous GROUP was ended.

error: bad LU or device not supported

> An LU specified in the command is not a disk or tape device supported by FC.

error: bad destination name

> Illegal file name specified in the destination parameter.

error: bad file1 or file2 param

> The value for the file1 or file2 parameter was incorrectly specified.

error: bad msc parameter

error: bad namr

> An incorrectly formed namr appeared in the command.

error: bad or missing parameter

error: bad tape LU

> The tape LU specified in the command is not that of a tape device supported by FC.

error: braces used where not permitted

error: C and ! options require explicit destination cartridge

> C or ! option cannot be specified without also specifying the destination disk cartridge explicitly.

error: comment file checksum error

> A checksum error was detected when reading the comment file from the tape.

error: dest secu code not allowed w/o source secu code or good msc

> A security code was specified in the destination parameter, but not in the source parameter, and the msc parameter was not specified correctly.

FC Errors (continued)

error: error on tape write

An unrecoverable error was reported by the driver on a tape write request.

error: fatal i/o error on LU nnn, status=<octal A-Register status>

FC attempted I/O to the specified LU, and the status returned in the A-Register indicates a problem that should never occur.

error: file name BOOTEX reserved for system

Destination parameter specified BOOTEX as the destination file name, which is not permitted.

error: in cartridge status on disk LU nnn:
FMGR <nnn>

Same as "cartridge lock error...", but more general.

error: incompatible options

Contradictory options were specified in the same command.

error: L and O options must be consistent for a given cartridge

error: L and O options require explicit source cartridge

L or O option cannot be specified without also specifying the source disk cartridge explicitly.

error: mismatched braces

error: namr list not allowed in this context

A list of namrs delimited by braces was used in a parameter where not permitted.

error: no such command (use ? for help)

error: no tape-to-tape copy

Both the source and destination parameter specified a tape device.

error: not enough memory
error: memory overflow
error: out of memory

FC does not have enough free memory to allow the command to proceed.

error: not in group

An EG (END GROUP) or AG (ABORT GROUP) command was entered, but there was no previous GROUP to end or abort.

error: number of cartridges to be copied to tape exceeds limit

Cartridges were renamed on a copy to tape such that tape's cartridge list would contain more than 64.

FC Errors (continued)

error: on disk-to-tape copy, dest cartridge cannot be specified as −LU

**error: on help file "FCHLP:
FMGR <nnn>**

Error nnn occurred when attempting to access FC's help file, "FCHLP, needed for the ? command.

error: option not applicable

An option character was specified in a COPY command that is not applicable for this type of copy.

error: option not defined for this command

A character in the "options" parameter is not meaningful for this command.

error: options inconsistent with group

Within a GROUP, certain options (V, I, S, F, B, T, and K) cannot be specified in one copy command but omitted from another.

error: P option not allowed if source and dest are same cartridge

P option cannot be specified on a disk-to-disk copy in which source and destination are on the same cartridge.

error: P option requires explicit source cartridge

P option cannot be specified without also specifying the source disk cartridge explicitly.

error: renaming multiple files to same name, or cartridge not found

The source name was omitted, or has wildcards, or the source is a list of namrs but the destination specifies a file name; the destination parameter was intended to specify an ASCII CRN in the abbreviated form (CRN rather than ::CRN) but the ASCII CRN was interpreted as a file name because no cartridge with that CRN was found on the cartridge list.

error: source or destination incompatible with group

The source and destination parameters must be consistently either a disk or a tape throughout the GROUP, and only one tape LU can be specified in a given group.

error: tape DESCRIBE error

CTD driver reported an error when FC issued a DESCRIBE control request.

error: tape lock failed

FC is unable to lock the tape LU required by the command.

FC Errors (continued)

error: unrecognized option character

An unrecognized character appeared in the "option" parameter.

error: volume not big enough for header/comment/directory files

Destination tape too small.

exec error <system error mnemonic>, exec params: <params in octal>

The operating system or I/O driver rejected an EXEC call made by FC. The numeric values are for diagnostic use only.

fatal scratch file error: FMGR <nnn>

FMGR error nnn occurred when FC tried to create or access one of its scratch files.

fatal tape read error

The tape driver reported an error while the header or comment file was being read.

<source namr> [<dest namr>]
files lost, reference numbers <n1> thru <n2>

Distributed directory information was lost for files with reference numbers n1 through n2.

files lost, reference numbers <n1> thru <n2>, names not available

Distributed directory information was lost for files with reference numbers n1 through n2.

<source namr> [<dest namr>]
file too large for this operating system

Similar to "file copied but not purged..." error, except that the "file too large..." error occurred rather than a FMG error.

FMGR <nnn>

When a namr or a pair of namrs is logged, followed by an FMGR error code, the indicated file was not copied because of the FMGR error, except for FMGR-001 errors on disk-to-tape copies. For the FMGR-001 error, the file is copied to tape but the affected parts of the file are flagged to indicate the disk error.

A single namr is the source file; a pair of namrs are the source and destination files.

<source namr> [<dest namr>]
FMGR <nnn>
warning: above file copied but not purged from source

Similar to the above error, except that the FMGR error only prevented the source file from being purged as requested by the P option, and did not prevent the file from being copied.

FC Errors (continued)

group aborted

The AG (ABORT GROUP) or AB (ABORT FC) command was entered, causing all COPY commands since the GROUP command to be ignored.

header file checksum error

The information in the header file is questionable due to a checksum error.

internal error: <10-character code>

FC detected an inconsistency that cannot be interpreted; code is for diagnostic use only.

internal error: pascal <error type#> <error#> <line#>

FC is being terminated due to a Pascal run-time error; numeric values are for diagnostic use only.

<source namr> [<dest namr>]
I/O– <error code> on LU nnn[,D] [,F]

Same meaning as the operating system error message
'**I/O– <error code> @LU nnn[,D][,F]

list file error:
FMGR <nnn>

FMGR error nnn occurred when FC attempted to access the list file (determined by the most recent LL command).

LU nnn is down

correct problem and UP device, or use BR to break FC.

no files selected

The COPY command had no effect because no source files were selected.

possible data loss:
[xtnt <x1>] blk <b1> thru [xtnt <xn>] blk <bn>

Same as "possible data loss: entire file," except that only part of the file might have been lost.

<source namr> [<dest namr>]
possible data loss: entire file

Same as "data lost: entire file," except that it is uncertain whether any data was lost.

file <name>::<CRN>
purged or replaced during copy or directory entry corrupt

The indicated file was purged and possibly replaced by a different file during the time FC was scanning the directory, or else a directory entry for the indicated file is corrupt.

FC Errors (continued)

<source namr>[<dest namr>]
put volume [number nnn] on LU <tape LU>
when ready to continue type GO, otherwise type BR [or SK]

Tape not mounted and on line, or loaded.

retrying tape read

A "checksum error..." or "bad record length" error, has occurred, and FC is attempting to recover from the error by retrying the tape read.

source file <name>::<CRN>
selected by commands with conflicting parameters

On a tape-to-disk copy specified by a GROUP with more than one COPY command, the indicated source file was selected by more than one COPY command, but those particular COPY commands did not meet the following restrictions:

1. The destination parameters must be the same in all the commands.

2. The D option must be consistently selected or not selected in all the commands.

3. The E option must be consistently selected or not selected in all the commands.

4. The msc parameter must be consistent through all the commands.

5. It is not permissible for the L option to be specified in some of the commands and the O option in others.

tape LU nnn media not initialized
when ready to continue type GO, otherwise type BR

An attempt was made to write to an uninitialized CTD cartridge.

tape LU nnn not ready
when ready to continue type GO, otherwise type BR [or SK]

Mag tape device is not on line or CTD cartridge is not loaded.

tape LU nnn not write enabled
when ready to continue type GO, otherwise type BR

Write-protect ring not installed on tape, or protect switch on CTD does not have arrow pointing away from the direction labelled SAFE.

tape format error <negative error code>

The data read from tape deviated from the expected format so severely that FC is unable to continue reading the volume; code is for diagnostic use only.

This error differs from a tape format error with a positive error code.

FC Errors (continued)

tape format error <positive error code>

The data read from tape deviated from the expected format in some way, but copying from the tape could continue.

tape not readable by FC, first two records are:
<record 1>
<record 2>

Tape mounted is apparently not an FC tape.

tape read error

The driver indicated an unrecoverable error on the tape read.

(timeout)

FC detected a terminal time-out while waiting for input from the log device.

TR stack overflow

volume does not match others

On a multivolume tape-to-disk copy, the tape volume just mounted does not come from the same set as the previous volumes.

waiting to lock list device

FC attempted to lock the list device and either another program had the list device locked, or a resource number for the lock was not available.

warning: no match for:[namr = <namr>][file1 = <?>][file2 = <?>]

No files were found to match the combined restrictions of the source namr and the file1 and file2 parameters.

warning: title truncated
title: <title>

The TITLE command specified a title that was too long.

warning: unable to eliminate extents in one or more files

The E option was specified, but extents could not be eliminated from some files.

wrong volume

The tape volume just mounted does not have the correct volume number.

FMP and FMGR Error Messages

ERROR	ERROR MESSAGE
060	DO YOU REALLY WANT TO PURGE THIS DISK?
057	BAD TRACK IS OUTSIDE FILE AREA
056	BAD PARAMETER
055	MISSING PARAMETER
053	ILLEGAL LABEL
052	ILLEGAL LU
051	ILLEGAL MASTER SECURITY CODE
050	NOT ENOUGH PARAMETERS
049	CAN'T RUN THIS PROGRAM
048	GLOBAL IS SET OUT OF RANGE
046	INSUFFICIENT CAPABILITY
042	LU CANNOT BE SWITCHED
039	CAN'T RP THIS PROGRAM
038	CAN'T REMOVE ACTIVE PROGRAM OR SWAP FILE
037	DEVICE IS BUSY
036	NO TIME VALUES GIVEN
023	DUPLICATE PROGRAM NAME
022	COPY TERMINATED
021	ILLEGAL DISK SPECIFIED
020	ILLEGAL TYPE ZERO FILE
019	PROGRAM IS NOT LINKed FOR THIS SYSTEM
018	PROGRAM IS NOT DORMANT
014	ID SEGMENT NOT FOUND
013	TRANSFER FILES ARE TOO DEEPLY NESTED
012	DUPLICATE DISK LABEL OR LU
010	INPUT ERROR
007	CHECKSUM ERROR

FMP and FMGR Error Messages (continued)

ERROR	ERROR MESSAGE
006	FMGR SUSPENDED
001	DISK ERROR–LU REPORTED
000	BREAK
–000	(no error)
–001	DISK ERROR!
–002	FILE ALREADY EXISTS
–003	BACKSPACE ILLEGAL
–004	RECORD SIZE ILLEGAL
–005	BAD RECORD LENGTH
–006	NO SUCH FILE
–007	INCORRECT SECURITY CODE
–008	FILE IS ALREADY OPEN
–009	MUST NOT BE A DEVICE
–010	NOT ENOUGH PARAMETERS
–011	DCB IS NOT OPEN
–012	ILLEGAL FILE POSITION
–013	DISK IS LOCKED
–014	DIRECTORY IS FULL
–015	ILLEGAL NAME
–016	SIZE=0 OR ILLEGAL TYPE 0 FILE ACCESS
–017	DEVICE I/O FAILED
–018	ILLEGAL LU.
–032	NO SUCH CARTRIDGE
–033	RAN OUT OF DISK SPACE
–034	DISK IS ALREADY MOUNTED
–035	ALREADY 63 DISKS MOUNTED TO SYSTEM
–036	LOCK ERROR ON DEVICE
–037	PROGRAM IS ACTIVE
–038	ILLEGAL SCRATCH FILE NUMBER

FMP and FMGR Error Messages (continued)

ERROR	ERROR MESSAGE
−046	GREATER THAN 255 EXTENTS
−049	COPY VERIFY FAILED
−050	NO FILES FOUND
−051	DIRECTORY IS EMPTY
−053	PROGRAM ASSIGNED TO BAD PARTITION
−054	PARTITION TOO SMALL FOR PROGRAM
−055	NO ROOM IN SHAREABLE EMA TABLE
−056	SHEMA ASSIGNED TO NON−EXISTENT PARTITION
−057	PARTITION TOO SMALL FOR SHAREABLE EMA
−058	PROGRAM OR DATA PARTITION ASSIGNED TO SHEMA PARTITION
−059	255 PROGRAMS USING SHAREABLE EMA AREA
−060	CODE AND DATA ASSIGNED TO SAME PARTITION
−063	CODE ASSIGNED TO NON−EXISTENT PARTITION
−064	PARTITION TOO SMALL FOR CODE SEGMENT
−068	CODE ASSIGNED TO SHAREABLE EMA PARTITION
−099	D.RTR EXEC REQUEST ABORTED
−101	ILLEGAL PARAMETER IN D.RTR CALL
−102	D.RTR NOT AVAILABLE
−103	DIRECTORY IS CORRUPT
−104	MISSING EXTENT
−105	D.RTR MUST BE SIZED UP
−108	ILLEGAL NUMBER OF SECTORS/TRACK
−200	NO WORKING DIRECTORY
−201	DIRECTORY NOT EMPTY
−202	DID NOT ASK TO READ
−203	DID NOT ASK TO WRITE
−204	FILE READ PROTECTED

FMP and FMGR Error Messages (continued)

ERROR	ERROR MESSAGE
−205	FILE WRITE PROTECTED
−206	DIRECTORY READ PROTECTED
−207	DIRECTORY WRITE PROTECTED
−208	DUPLICATE DIRECTORY NAME
−209	NO SUCH DIRECTORY
−210	UNPURGE FAILED
−211	DIRECTORIES NOT ON SAME LU
−212	CANNOT CHANGE THAT PROPERTY
−213	TOO MANY OPEN FILES
−214	DISK NOT MOUNTED
−215	TOO MANY DIRECTORIES
−216	YOU DO NOT OWN
−217	BAD DIRECTORY BLOCK
−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 IS BUSY
−226	PROGRAM 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 FMGR FILE
−233	NO SUCH USER

Computer
Museum

FMP and FMGR Error Messages (continued)

ERROR	ERROR MESSAGE
-234	SIZE MISMATCH ON COPY
-235	BREAK FLAG DETECTED
-236	YOU ARE NOT A SUPERUSER
-237	MUST NOT BE REMOTE
-238	ILLEGAL PROGRAM FILE
-239	PROGRAM NAME EXISTS
-240	CHANGED RPL CHECKSUM
-241	CAN ONLY RUN UNSHARED
-242	DISK I/O FAILED
-243	PARAMETER ERROR
-244	MAPPING ERROR
-245	SYSTEM CAN'T DO CDS
-246	SYSTEM COMMON CHANGED
-247	UDSP NOT DEFINED
-248	INVALID DIRECTORY ADDRESS FOUND
-249	SECURITY VIOLATION DETECTED
-250	D.ERR NOT AVAILABLE
-251	PROGRAM NAME EXISTS IN ANOTHER SESSION
-252	DISK LU IS DOWN
-253	DISK LU IS LOCKED
-254	NO SUCH GROUP
-255	USER IS NOT IN GROUP
-256	NO SUCH SESSION
-257	NO SUCH PROGRAM
-258	NO SAM FOR PROTO ID
-260	TOO MANY SYMBOLIC LINKS IN PATH
-261	SYMBOLIC LINK RESULTS IN ILLEGAL PATH
-262	SYMBOLIC LINK MUST NOT BE REMOTE
-263	SYSTEM DOES NOT SUPPORT SYMBOLIC LINKS

FMP and FMGR Error Messages (continued)

ERROR	ERROR MESSAGE
−270	UPDATE TIME ALREADY CURRENT
−300	ILLEGAL REMOTE ACCESS
−301	TOO MANY REMOTE CONNECTIONS
−302	NO SUCH NODE
−303	TOO MANY SESSIONS
−304	NO SUCH ACCOUNT
−305	INCORRECT PASSWORD
−306	CAN'T ACCESS ACCOUNT
−307	TRANSFER IS TOO LONG
−308	CONNECTION BROKEN
−310	DS IS NOT INITIALIZED
−311	DS IS NOT CONNECTED
−312	REMOTE SYSTEM DOESN'T RESPOND
−313	NO TRFAS AT REMOTE SYSTEM
−314	INTERNAL TABLE FULL IN REMOTE TRFAS
−315	DS ERROR DSXX(X), NODE YY
−401	MESSAGE NUMBER NOT FOUND IN CATALOG
−402	MESSAGE TOO BIG FOR MESSAGE BUFFER

FORTRAN Compilation Errors

1 **Error in ftn directive.**

The first line of the source begins with "FTN" but does not conform to the syntax for the FTN directive or has an illegal option.

2 **Illegal option in runstring.**

The fifth parameter in the runstring contains a character that is not a legal option. The options I, J, X, Y, and E can appear in the FTN directive, but not in the runstring.

3 **Compiler space overflow; compiler needs more EMA memory.**

Refer to the installation manual for information on how to size the compiler.

4 **Invalid common label.**

The common label given has an illegal character or is not followed by a matching slash (/).

5 **Implicit is redundant or retypes named constant.**

A starting letter or range of letters is used in more than one IMPLICIT group, or a name in a PARAMETER statement has its implicit type changed by this statement.

6 **Transfer of control into a loop or IF-THEN-ELSE block.**

A statement in a loop or IF-THEN-ELSE block is referenced from outside the loop or block. The program may not run as expected. When control is transferred into a DO loop, the loop may not be initialized correctly; in an IF-THEN-ELSE, the block may have been optimized out.

7 **ENTRY or RETURN in main program or BLOCK DATA.**

These statements are legal only in subroutines and functions. If a SUBROUTINE or FUNCTION statement has an error, the compiler may consider this module to be a main program.

8 **Illegal complex number.**

The number looks like a complex constant, but does not conform to the rules for forming such constants.

9 **Mismatched parenthesis.**

There are either more right parentheses than left parentheses at some point, or not as many right parentheses as left parentheses at the end of the expression.

10 **Unrecognized statement.**

The statement does not look like an assignment statement or a statement function, and does not start with a recognized keyword.

FORTRAN Compilation Errors (continued)

11 `Dimension/substring 2nd part < 1st part.`

The upper bound or last character position is less than the lower bound or first character position.

12 `Return # too large or too many alternate returns.`

The (constant) return number in a RETURN statement exceeds the number declared in the SUBROUTINE statement, or a system intrinsic (for example, EXEC) call has more than one alternate return specified.

13 `Constant > 2047 in format, or illegal string.`

Numeric value in a FORMAT is too large, or the closing quote of a string is missing, or an ALIAS string is too long.

14 `Constant or constant expression overflow or under flow.`

A constant (expression) exceeds the number range of the machine, using the data types of the constants.

15 `Keyword unrecognized, repeated, or illegal.`

An I/O keyword is misspelled, has already been used in this I/O statement, or is illegal in this I/O statement.

16 `Illegal octal or hex constant.`

The constant has an illegal character or is too big.

17 `Missing constant or operand.`

An operand was expected but a delimiter or an unrecognized character was found. This can be caused in a number of ways by improper syntax or badly formed names or constants.

18 `Illegal combination of keywords.`

Some of the keywords used in this I/O statement cannot be used together.

19 `Integer constant expected.`

Only an integer constant, or possibly an integer constant expression, can be used in this context.

20 `Illegal character count in hollerith constant.`

A negative or zero integer constant is followed by "H". Hollerith constants must have character counts greater than zero, and the "H" is not legal here if Hollerith was not intended.

21 `Value out of range.`

The constant value given is too large, too small, or otherwise unacceptable.

FORTRAN Compilation Errors (continued)

22 **Illegal use of name.**

This name has been used before in a different context that conflicts with its current use.

23 **Step size = 0.**

The step size in a DO loop cannot be zero.

24 **Variable or array name expected.**

A variable or array name is expected here, but a delimiter, constant, named constant, or subprogram name is found.

25 **Variable name or constant expected.**

A name is expected here, but a delimiter, constant, named constant, or subprogram name is found.

26 **Integer (logical) item expected.**

The context calls for an item of a certain type (integer or logical) but the item given is of a different type.

27 **Duplicate statement number.**

This statement number has already been used.

28 **Unexpected character or unexpected end of statement.**

The compiler did not recognize the character, or was expecting more characters to complete the statement.

29 **Blank line has statement number.**

Blank lines should not have statement numbers; blank lines are comments.

30 **Incorrect nesting. May be due to other errors.**

If there are other errors, this error may be ignored until the other errors are corrected. Can be caused by END DO, ELSE, ELSE IF, and ENDIF statements after errors. It may also be caused by ending a block (for example, a DO loop) before all inner blocks have been ended.

31 **Compiler space overflow; compiler needs more (EMA) memory.**

Refer to the Configuration/Installation manual for information on how to size the compiler.

32 **Undefined, illegal or incorrectly used statement number.**

This statement number has an illegal character in it, was never defined, or was used in the wrong context (for example, a FORMAT statement number cannot be used in a GOTO).

FORTRAN Compilation Errors (continued)

33 Redundant/conflicting/missing EXTERNAL/INTRINSIC declarations.

A subprogram name was passed as an actual parameter but not declared EXTERNAL or INTRINSIC, or was declared more than once in EXTERNAL or INTRINSIC statements.

34 Statement out of order.

This statement appears too late in the source. This can be due to errors in earlier statements.

35 No path to this statement.

The statement before this one always transfers control (for example, a GOTO) but this statement is not labeled, so it can never be reached. If this happens in old code that uses GOTOs after EXEC calls with the no-abort bit, the calls *must* be changed to use alternate returns. The use of GOTO with the no-abort bit is *not* supported.

36 Variable appears twice in common.

A variable or array name appears more than once in COMMON statements.

37 Formal parameter in COMMON or DATA statement.

Parameters are illegal in this context.

38 Wrong number of subscripts.

This array reference has a different number of subscripts than were declared when the array was dimensioned.

39 Illegal variable in dimension bound expression.

Only formal arguments and variables in common may be used as dimension bounds or in bound expressions.

40 Inconsistent equivalence group.

It is impossible to perform the equivalences specified, because more than one location has been given for an item. Examples: Items in different common blocks; or two different elements of an array specified, but not aligned properly.

41 Negative extension of common via equivalence.

An array element was equivalenced to an item in common in such a way that the start of the array would be before the start of the common block.

42 Left parenthesis expected.

Probably a function reference without a parameter list. Parameter lists are required, even if empty; for example, PCOUNT( ).

FORTRAN Compilation Errors (continued)

43 **Variable used in a context that requires a formal parameter.**

Some operations, such as declaring an array to be variably dimensioned, are only legal for formal parameters.

44 **Constant missing, ill-formed, or of wrong type.**

The compiler could not recognize a constant of an acceptable type in this context.

45 **Illegal combination of data types.**

The current operation cannot be performed on the items given; the data types are not compatible.

46 **Name of a function not used or name of a subroutine is used.**

A function name must be used for assigning a result value; a subroutine name must not be used as a variable.

47 **Variable dimension bound not in same entry list as the array.**

The specified variable is a formal parameter but is not given in an entry list that contains the array; so the bound is undefined at that entry, and the array address calculations cannot be done.

48 **Illegal use of EMA variable.**

EMA variables are not allowed in some contexts, such as some I/O keyword values.

49 **Function has illegal mix of entry point types.**

Character and noncharacter entry points cannot be mixed, and character entry points must all have the same length.

50 **Illegal last statement of DO loop.**

This statement cannot end a DO loop. In particular, any statement that always causes a transfer of control is illegal here (for example, GOTO or arithmetic IF).

51 **Control variable of DO statement already in use.**

An outer loop uses the same index variable as this inner loop.

52 **This statement may not be used as the true part of a logical IF.**

DO, logical IF, block-IF, ENTRY and END statements cannot be used as the statement following a logical IF.

53 **Illegal use of character*(*) or "*" last upper bound.**

Character*(*) can only be used for a formal argument or named constant; "*" can only be used as the last upper bound of a formal argument.

FORTRAN Compilation Errors (continued)

54 Array name dimensioned twice.

Array names followed by their dimensions can only appear once within the specification statements.

55 Illegal use of non-character data.

Character data is required in this context.

56 Illegal combination of data types.

These data types cannot be used together with any operator.

57 Illegal data type.

This data type is illegal in this context (for example, with a certain operator, such as logical data with "+").

58 Function used as subroutine or has alternate returns.

This subprogram is used as a function, but it has alternate returns or is also used as a subroutine. Mixed use as a function and subroutine is allowed for ALIASed items, but should not be used in new programs.

59 Wrong # of arguments.

This intrinsic, statement function, or recursive call has an illegal or inconsistent number of actual arguments.

60 Illegal argument type.

This intrinsic, statement function, or recursive call has an argument with an illegal or inconsistent type.

61 Arithmetic IF with illegal data type.

The expression in an arithmetic IF must be numeric and not of type COMPLEX. Make sure the parentheses are balanced and the right side has the correct syntax.

62 Logical IF or WHILE with non-logical data.

The expression in a logical IF or DO WHILE must be of type LOGICAL. Make sure the parentheses are balanced and the right side has the correct syntax.

63 PCOUNT used with DIRECT entry point.

The number of actual parameters passed to a DIRECT entry point cannot be determined.

64 "THEN" expected.

The only legal statement after ELSE IF (*expression*) is THEN.

65 Non-character item on odd byte address.

A COMMON or EQUIVALENCE statement puts a word-addressed variable on an odd byte address. In a COMMON statement, a byte will be left unused.

FORTRAN Compilation Errors (continued)

66 `Program should (not) have executable statements.`

Regular subprograms and main programs must have executable statements, and BLOCK DATA subprograms must not have them.

67 `Character item cannot be in EMA.`

By explicit declaration, COMMON, or EQUIVALENCE, an attempt was made to put character data in EMA, which is illegal.

68 `ENTRY used in Block IF or DO loop.`

All ENTRY statements must be at the "outer" level of a subprogram, and not within any DO or IF blocks.

69 `Substring out of bounds or non-character item has substring.`

In a character item, the starting character is less than one or the ending character is greater than the declared length.

70 `Constant subscript out of bounds.`

The subscripts are outside of the declared dimensions.

71 `Too many/few constants, or illegal repeat count.`

The number of constants does not match the number of variables and array elements, or the repeat count is negative or zero.

72 `Item must (not) be in common.`

In a BLOCK DATA subprogram, only items in labeled common can be used in DATA statements.

73 `Constant & variable have different types.`

This constant cannot be converted to the type of the corresponding variable. Make sure the number of variables and constants is correct up to this point.

74 `Undeclared array, or stmt funct after first executable stmt.`

This statement looks like an assignment to an array element or like a statement function, but the name is not declared as an array and it is too late in the program to define a statement function.

75 `Illegal use of current subprogram name or ENTRY name.`

Entry point names are illegal in this context (for example, a recursive call in non-CDS mode).

76 `Duplicate formal parameter.`

A formal parameter appears more than once in this parameter list.

77 `Statement number ignored.`

Statement numbers are not functional on specifications, DATA, or ENTRY statements. The number is ignored, as if the statement number field had been blank.

FORTRAN Compilation Errors (continued)

78   Overlays (type-5 programs) not allowed in CDS mode.

The module has been converted to a zero-argument subroutine.

79   More than 255 parameters to a subprogram or library routine.

The (CDS) PCAL instruction can only handle calls with 255 or fewer parameters. The offending call can be caused by certain long expressions, such as concatenations.

80   This statement not legal in BLOCK DATA.

Only certain specifications and data statements are legal in BLOCK DATA; all executable statements and formats are illegal.

81   PROGRAM formal parameter is not type character.

All formal parameters in a PROGRAM statement must be of type CHARACTER.

82   Compiler space overflow; compiler needs more memory or EMA.

There was not enough memory available for the compiler to compile this module. Increase the "size" of the compiler. See your System Manager, who should refer to the Configuration/Installation manual for information on how to size the compiler.

83   Attempt to retype a name.

A name appears in more than one type statement or twice in a single type statement.

84   Code, data, common, save, stack or EMA space overflow.

The module is too big. If there are large arrays, try making them smaller or moving them to EMA. If the module is very long, try breaking it into two smaller modules.

85   Program name conflicts with common, external or intrinsic name.

The program name should be unique. Most "internal" names used to access the library contain special characters, but a few do not, such as SIN. Choose another program name.

86   (not currently used)

87   The given names were not typed.

IMPLICIT NONE was specified but some variables were not mentioned in type statements. Could be due to typographical errors.

88   Cannot access list file.

An OPEN, CREATE, or WRITE of the list file failed. If the list file does not end with .LST (new files) or begin with a single quote (old files), it cannot be overwritten. If the list file is given the default name using the "−" convention, the source file name must end with .FTN (new files) or begin with an ampersand (old files).

FORTRAN Compilation Errors (continued)

89   (not currently used)

90   `Illegal continuation line.`

The first line of the program, or the first line after a directive, cannot be continued.

91   `The following external name is used in more than one way.`

A subprogram name was used as an entry point or a common block name, or was ALIASed to match another subprogram name.

92   `External name conflicts with a library routine.`

An entry point, common block, or subprogram name matches the "internal" name used for a library routine. Most "internal" names contain special characters, but a few do not, such as SIN. Choose another name.

93   (not currently used)

94   `Item cannot be declared in EMA statement or is declared twice.`

Only formal parameters and local variables can be declared to be in EMA, and must appear only once in EMA statements.

95   `Cannot open include file, or name greater than 63 chars.`

The OPEN of this include file failed. Check the name.

96   `Break detected.`

An operator BREAK was entered. The compilation was terminated. The relocatable file is not usable, and the list file may be incomplete.

97   `Cannot access relocatable output file.`

An OPEN, CREATE or WRITE of the relocatable file failed. If the relocatable file does not end with .REL (new files) or begin with a percent sign (old files), it cannot be overwritten. If the relocatable file is given the default name using the "−" convention, the source file name must end with .FTN (new files) or begin with an ampersand (old files).

98   `Cannot access source file, or eof before end.`

The end of the source file was encountered in the middle of a module (because of a missing or unrecognized END), or an OPEN or READ of the source file failed, or a READ of an included file failed.

99   `Can't access scratch file(s).`

A CREATE or WRITE of internal scratch files failed, probably due to lack of space. Make sure there is space available on the scratch cartridge, working directory, or top cartridge, depending on your operating system. If in doubt, consult the System Manager.

FORTRAN Compilation Errors (continued)

100 (not currently used).

101 $IF/$ELSE/$ENDIF nested incorrectly, or nested too deeply.

These directives were nested to more than 16 levels, or incorrectly nested.

102 Command-line directive must start with $; must not be INCLUDE.

The sixth command-line argument on the FTN7X command line did not start with $ or was $INCLUDE.

103 Ill-formed {name} reference to $SET variable.

A left brace "{" was not followed by a name and a right brace "}".

104 No such $SET variable.

This name has not been defined in a $SET directive yet.

105 $SET used between $ALIAS or $EMA and start of subprogram.

$SET may not be used between $ALIAS or $EMA and the start of the following subprogram.

106 Character item used on both sides of assignment; must not overlap.

A character variable, array, or substring was used on the left side of an assignment and in the expression on the right side. It is possible that the two usages overlap, which could produce an incorrect result or runtime error. The compiler can only determine the possibility of overlap. See the P option.

107 Variables in absolute common blocks may not appear in DATA statements.

When $ALIAS is used to place a common block at an absolute address, variables in that common block may not be initialized in DATA statements.

FORTRAN Run-Time Errors

The input/output run-time errors returned in the IOSTAT variable or displayed on the user's terminal are:

Input/Output Run-Time Errors

−1   An EOF was read on a sequential file, or the end of an internal file was reached.

450   Invalid FORTRAN unit number (less than zero) or system unit number (greater than 255).

451   Unrecognized STATUS value; legal values in an OPEN statement are OLD, NEW, SCRATCH, and UNKNOWN.

452   No file name (FILE=) given; file names are required when STATUS is OLD or NEW.

453   File name (FILE=) supplied; no file names are allowed when STATUS is SCRATCH.

454   Unrecognized ACCESS value; legal values are SEQUENTIAL, DIRECT, and BLOCKS.

455   Unrecognized FORM value; legal values are FORMATTED and UNFORMATTED.

456   The value for MAXREC, RECL, BUFSIZE, or all is negative or zero. These parameters must have positive values.

457   Unrecognized BLANK value; legal values are NULL and ZERO.

458   The maximum number of scratch files, 99, were in use, so this OPEN of another scratch file could not be done.

459   This file has already been opened and connected to a different unit; a file can be connected to only one unit at a time.

460   The OPEN specified direct access, but the file to be opened was sequential access (not type 1 or 2).

461   The OPEN specified sequential access, but the file to be opened was direct access (type 1 or 2).

462   The file was not found, and STATUS was OLD. Same as error 506.

463   Unrecognized STATUS value; legal values in a CLOSE statement are KEEP and DELETE.

464   An ENDFILE was attempted on a direct access file; such files do not have EOFs and ENDFILE is illegal.

FORTRAN Run-Time Errors (continued)

465   Invalid file name given in FILE=. Same as error 515.

466   All connections specified in the $FILES are in use; no more
       OPENs can be done until a CLOSE is done.

467   All disk file connections specified in the $FILES directive are
       in use; no more disk file OPENs can be done until a disk file
       CLOSE is done.

468   The record length given in RECL does not match the actual
       record length of the file.

469   The file position given in FFLOC is odd; only even byte posi-
       tions are allowed, because the file system cannot position a
       file to an odd byte position.

470   Unrecognized USE value; legal values are EXCLUSIVE,
       NONEXCLUSIVE, and UPDATE.

471   The system unit number used is not accessible from this pro-
       gram; that is, it is not in the session SST.

472   Could get neither read nor write permission for this file.

473   (not currently used)

474   A record number (REC=) was supplied in the READ or
       WRITE, but the unit was not connected to a direct access file.

475   A RECL value must or must not be supplied. RECL must be
       used *only* if ACCESS is DIRECT, or BLOCKS.

476   (not currently used)

477   A NODE value was supplied (other than -1), but the $FILES
       directive did not contain the DS keyword (FMGR file system
       only).

478   An OPEN of a unit that was already connected tried to
       change attributes other than BLANK.

479   An OPEN was tried with $FILES 0,0; or the library routines
       to support OPEN were not loaded correctly.

480   A CLOSE was tried with $FILES 0,0; or the library routines
       to support CLOSE were not loaded correctly.

481   An INQUIRE was tried with $FILES 0,0; or the library rou-
       tines to support INQUIRE were not loaded correctly.

482   The library routines to support BACKSPACE, ENDFILE,
       and REWIND were not loaded correctly.

FORTRAN Run-Time Errors (continued)

483   This INQUIRE statement tried to inquire about a disk file, but the $FILES directive did not specify any disk connections. At least one disk connection must be specified, even if no OPEN of a disk file will be done.

484   This OPEN statement tried to open a disk file, but the $FILES directive did not specify any disk connections.

485   This OPEN statement specifies a direct access file or uses RECL, but the file name supplied is an LU number.

486   Attempt to use DNODE, which is illegal in FORTRAN 77 programs. Use an OPEN statement to connect to a remote unit.

487   ZBUF, ZLEN, or secondary/tertiary address supplied in this READ or WRITE statement, but unit is connected to a disk file.

488   REC supplied in this READ or WRITE statement is negative.

489   (not currently used)

490   (not currently used)

491   This FORMAT has an invalid field width (w), number of digits (d), minimum number of digits (m), or size of exponent output (e).

492   This FORMAT does not begin with a left parenthesis, or has too many levels of parentheses.

493   Unrecognized format character, or use of a negative value in a format (except scale), or no conversions given in the format but I/O list was not empty.

494   Illegal character in a numeric input field.

495   Numeric input field has an ill-formed number or logical value, or an octal number is too large.

496   Discrepancy in record size, I/O list length, internal buffer size, or all. Could be due to:

FORTRAN Run-Time Errors (continued)

- Input record (or amount that fits in internal buffer) was not large enough to satisfy an unformatted READ list. If unknown-length records are being read with an unformatted READ, the error can be trapped with IOSTAT and the actual record length recovered using ITLOG.

- Output record, as specified in unformatted WRITE list or a FORMAT, was too large to fit in the internal buffer. See library routine LGBUF in Appendix B.

- Output record, as specified in unformatted WRITE list or a FORMAT statement, was larger than the record size for the direct access file to which this unit is connected.

497   A format specifier is illegal for the type of the list item that matches it.

FMP Errors and DS FMP Errors

The following are the FMP and DS FMP errors. The last one to two digits of the error number correspond to the absolute value of the FMP number. For example, if the FMP error number is -6, the run-time error number is 506. If the FMP error number is -16, the run-time error number is 516.

For error messages that do not appear in this table, refer to the appropriate system reference manual for applicable system error messages and information about FMP, DS, and EXEC.

501   Disk error.

502   Duplicate file name.

504   Too many records (more than $2^{31}-1$) in a type 2 file in RTE-6/VM or RTE-A.

505   Record length illegal.

506   File not found.

507   Illegal security code or illegal WRITE to LU 2 or LU 3.

508   File OPEN or LOCK rejected.

512   EOF or SOF error.

513   Cartridge locked.

514   Directory full.

515   Illegal file name.

FORTRAN Run-Time Errors (continued)

516   Illegal file type.

519   Illegal access on a system disk.

525   Bad FCODE (internal RFAM error).

526   Bad entry number in RFAM; DCB destroyed.

528   Too many open DS files at remote node.

529   Internal RFAM tables invalid.

530   Disk not mounted to caller's session.

532   Cartridge not found.

533   No room on cartridge.

540   Disk not in SST.

541   No room in SST.

546   Greater than 255 extents.

547   No session LU available for SPOOL file.

Other error numbers in the range 501–999 (except 600/601) are unexpected FMP errors. FMP error number is ($500 - FTNerror\#$); for example, FORTRAN error 730 is really FMP error -230.

Character String Errors

The following are the character string errors:

600   Left and right sides of character assignment overlap (not an I/O error).

601   Substring out of bounds (not an I/O error).

I/O Errors

The following are the I/O errors. The last 2 digits of the error number correspond to the I/O error number. For example, if the I/O error is I005, the run-time error number is I005.

1000   An illegal class number was specified. Outside table, not allocated, or bad security code.

1001   Not enough parameters were specified.

1002   An illegal logical unit number was specified.

1003   Illegal EQT referenced by LU in I/O call (select code=0).

FORTRAN Run-Time Errors (continued)

1004   An illegal user buffer was specified. Extends beyond RT/BG area or not enough system available memory to buffer the request.

1005   An illegal disk track or sector was specified.

1006   A reference was made to a protected track or to unassigned LG tracks.

1007   The driver has rejected the call.

1009   The LG tracks overflowed.

1010   Class get call issued while one all already outstanding.

1011   A type 4 program made an unbuffered I/O request to a driver that did not do its own mapping.

1012   An I/O request specified a logical unit not defined for use by this session. The format for IO12 is:

SES LU =*xx* IO12 PROG ADDRESS:

where: xx = session LU not in SST

1013   An I/O request specified an LU that was either locked to another program or pointed to an EQT that was locked to another program.

1014   An I/O request was issued with the no-suspend option.

1015   Buffer size of a type 6 program is greater than what will fit in the user map.

1016   CPU backplane failure or I/O extender timing failure.

1020   Read attempted on write-only spool file.

1021   Read attempted past end-of-file.

1022   Second attempt to read JCL card from batch input file by other than FMGR. Revise program and rerun.

1023   Write attempted on read-only spool file.

1024   Write attempted beyond end-of-file; usually, spool file overflow.

1025   Attempt to access spool LU that is not currently set up.

1026   I/O request made to a spool that has been terminated by the GASP KS command.

FORTRAN Run-Time Errors (continued)

DS I/O Errors

The following are the DS errors that can occur during remote FMGR file access. These errors are generated only when your program was linked using the $FOLDF library and you used the $FILES directive with the DS option. The 11xx errors are generated when the FILE specifier in the OPEN statement is an LU number; the corresponding numbers in parenthesis (55x) are generated when the FILE specifier is a FMGR file name.

The last digit of the error number corresponds to the DS error number. For example, if the DS error is DS02, the error number is 1102 (552). For detailed DS error information, refer to the *DS/1000-IV User's Manual*, part number 91750-90012, or the *NS-ARPA Error Message and Recovery Manual*, part number 91790-90045.

1100 (550)	Local node is quiescent.
1101 (551)	Communication line parity, protocol failure, 'STOP' received, cable disconnected, or other hardware error.
1102 (552)	Communication line timeout error (DVA65 links only).
1103 (553)	Illegal record size.
1104 (554)	Illegal nodal address, node address not in nodal routing vector (NRV).
1105 (555)	Request timeout.
1106 (556)	Illegal request or monitor not active.
1107 (557)	System table error.
1108 (558)	Remote busy or resource unavailable.
1109 (559)	Illegal or missing parameters.

Miscellaneous Run-Time Errors

The following are miscellaneous run-time errors:

xx99	Partially malformed error messages.
1999	All RTE errors that do not have the form IOxx or DSxx and that are not FMP errors are mapped to 1999.
1000–1099	RTE I/O errors IO00 through IO99.
1100–1199	DS errors DS00 through DS99.

Halt Error Codes

When a halt is executed, the Virtual Control Panel (VCP) is invoked. It displays the message:

PXXXXXX AXXXXXX BXXXXXX MXXXXXX T1020zz

where:

P,A,B,M,T   are registers.

XXXXXX   are the contents of the registers.

zz   is the halt number, meaning:

00   The generated system has not been initialized by the BOOTEX program. Also displays the following message:

INITIALIZE SYSTEM BY RUNNING BOOTEX/BUILD

1   The system has not been initialized by the BOOTEX program. Can't mount disk; can't open boot cmd file; illegal use of shared partition for SHEMA. BOOTEX will usually try to print an error message prior to HLT 1.

2   A privileged routine has executed location 2. This can occur if data is executed or a link is set to 0. Problem with snap or system file (can't open, read/write error).

3   A Group II or III error has occurred when no program was executing.

4   A CPU power failure has occurred and the power fail driver was not included in the system.

5   A CPU parity error has occurred in the system or system common areas of memory.

11−17 (octal)

An interrupt has occurred from one of the select codes 11B to 17B.

Illegal Interrupt Error Codes

When an interrupt occurs on a channel for which there is no driver, the interrupt flag is cleared, and the message

**ILL INT−SCxx

where:

xx   is the select code of the interrupt source

is printed on the system console. No programs are affected by this error.

At generation time, a program can be set up to handle an interrupt directly. Whenever the interrupt occurs, RTIOL attempts to schedule the program. If the program is already busy, the message

INT−progr BUSY

where:

progr   is the name of the program

is printed on the system console. No programs are affected by this error.

INSTL Error Messages

LU XXXX IS NOT A DISK LU

The destination file can be created only on a disk. The program terminates after the error is found.

BOOT FILE NOT TYPE 1

The destination file must be type 1. The program terminates after the error is found.

DESTINATION FILE TOO SMALL

The destination file must be at least as large as the source. The program terminates after the error is found.

INSTL END. AAAAAA IS YOUR BOOT EXTENSION FILE.
WARNING: BOOT FILE MUST BE AT CYL 0, SECT 0.

These two messages are always given at the successful completion of the program when the destination is a file whether or not this condition is found.

INSTL END. YOUR BOOT EXTENSION HAS BEEN INSTALLED AT BOOT BLOCK XX ON LU YY.

This message is given at the successful completion of the program when the destination offset is a boot block number.

LINK Error Messages

Fatal LINK error messages have the form:

> message
> additionalInformation
> Fatal error nnn – Link terminated

Recoverable LINK error messages have the form:

Warning – nnn – message

nnn   3-digit number, over 100.

message   Brief error description. For more information refer to Appendix A of the LINK User's Manual.

additionalInformation

         Supplementary error information.

OLDRE Error Messages

OLDRE does not halt on an error, but continues translation of the entire file.

OLDRE: End OLDRE – No errors

Translation process successful.

OLDRE: End OLDRE – Relocatable file unchanged.

Translation process failed.

OLDRE: Input must be relocatable file – type 5

Relocatable file not type 5; OLDRE exits.

OLDRE: Relocatable must be a disk file

The file does not reside on a mounted cartridge; OLDRE exits.

OLDRE also aborts, with an appropriate message if a bad record checksum is encountered, the break flag is set, or a FMGR error occurs.

OLDRE: Converting new relocatable records to old format

This message displays while OLDRE is running.

OLDRE: Symbol name truncated to 5 characters
OLDRE: Local EMA, SAVE and code area illegal
OLDRE: New feature not available in old format
OLDRE: New feature in RPL illegal in old record
OLDRE: Cannot allocate EMA or SAVE common
OLDRE: More than 255 externals found

Any of these messages also can be displayed to describe conditions encountered during the translation process. They define errors that nullify the translation.

Parity Error Codes

CPU parity errors in the mapped part of memory (in a partition, not in the system common nor in the system area), cause the system to abort the program that encountered the parity error and issue the following message:

PE PAGE# XXXXXX OFFSET YYYYYY
name ABORTED PE address segment
DOWNED PAGES = n

where:

XXXXXX Page number where parity error occurred.

YYYYYY Offset of page.

name   Name of program.

address   Current program counter.

segment Number of segments most recently loaded (0 for nonsegmented programs).

n   Number of pages found to contain hard parity errors.

RTAGN Errors

Background swap priority error

Value not in range 0 through 32767.

Bad parameter

Parameter not of correct type or out of range.

Base page overflow

Blank common command out of order

Blank common command not in proper sequence in command file.

Buffer limit error

Buffer limit parameter in DVT command not of form BL:BU:ll:ul or BL:UN:ll:ul, with ll < ul, or not in range 0 through 32767.

Cartridge directory error

Incorrect format, or command not used (must be used, even if no disk in system).

Checksum error

File is not a relocatable file, is corrupt, or is incorrect type.

Class number specification error

The command to specify class numbers occurred out of order in the command file or was specified incorrectly.

Common usage not allowed

Modules in the operating system cannot specify common.

Device priority error

Priority specification in DVT command not of form PR,priority, or priority not in range 0 through 63 (77B).

Device type error

Device type parameter in DVT command not of form DT,deviceType, or deviceType not in range 0 through 63 (77B).

Driver parameter area size error

Driver parameter area size not specified before driver parameters, or specification not in range 0 through 511.

Driver parameter error

Driver parameter is out of range or more parameters specified than allowed for with the DX parameter.

RTAGN Errors (continued)

Driver partition size exceeded limit

Driver partition exceeds 8 pages.

Driver not partitionable

Driver relocated in partition does not have partitionable record in source. (Warning only.)

Duplicate entry points

Two subroutine entry points are the same.

DVT command or parameter error

The command to specify DVTs occurred out of order in the command file, or was specified incorrectly.

EMA usage not allowed

Expected "Node" command

The command to specify the NODE lists occurred out of order in the command file, or was incorrectly specified, or an LU in the list was in another list.

Fixup table overflow, size up

Generator requires a larger partition to relocate the modules. (Use SZ Command.)

Generator aborted

File already exists and may not be overlaid, or incomplete command file.

ID segment specification error

Command not of format ID,numSegs, or numSegs is nonnumeric or ≥ 0.

IFT command or parameter error

The command to specify IFTs occurred out of order in the command file, or was specified incorrectly, or its entry point was not found.

Illegal bound

LOCC or BLOCC is illegal.

Illegal driver name

Driver name not legal entry point name, or driver relocatable file not specified correctly.

RTAGN Errors (continued)

Illegal LU number

LU number not a positive integer in range 1 through 255.

Illegal model number

Model number not a positive integer less than 32767.

Illegal module name

Module name must not be numeric.

Illegal record or checksum error

Record in the file is of incorrect format or is corrupt.

Interface type error

The interface type specified for this IFT is not numeric or not in the range 0 to 63 (77B).

Internal table overflow, size up

Interrupt table command or parameter error

INT command to specify interrupt table entries was out of order in the command file, or was specified incorrectly.

Link error, current page used

LINK,BP or LINK,CP not specified.

LU specification error

LU out of range 0 to 255, or already assigned to a DVT.

Memory overflow

Relocated system is too large.

Missing system entry point

System entry point required by generator was not found.

Model number does not match driver

Incorrect model number subparameter, or file namr.

No DVT entry or no IFT entry

DVT or IFT entry missing or out of order.

Partition specification error

Illegal parameter in RS command.

Privileged interrupt error (fatal)

Interrupt table specification for privileged entry point is incorrect.

RTAGN Errors (continued)

Queuing error

Queuing option for IFT or DVT not of form QU:PR or QU:FI.

Record out of sequence

Tried to relocate a module not beginning at the first record of the module. Relocatable file is corrupt.

Resource number specification error

The RESN command to specify resource numbers occur- red out of order in the command file, or was specified incorrectly.

SAM specification error

SAM size specification number not a positive integer.

Segmented program not allowed

Attempt to include a segmented module in a generation relocation.

Segment RTAGn not found

Segments RTAG1−4 are not properly RPd.

Select code error

Select code has already been assigned to an IFT, is nonnumeric, or not in range 20B − 50B.

Shared program specification error

Shared program specification must be positive and nonzero.

Snap entry duplicate when truncated

All snap file symbols not unique in first five characters.

Snapshot file error

A read error occurred on the snap file that is created by the generator. Try again.

Specified module not found

Module not in file specified by RE, SE, or MS, or command not in proper format.

Spool limit specification error

Spool limits must be numeric.

Symbol table overflow, size up

Generator requires larger partition to relocate modules. Use SZ command.

RTAGN Errors (continued)

Table extension error

Table extension parameter of DVT command not of form
TX:numWords, or numWords not in range 0 through 511.

Time out value error

Time-out parameter in DVT command not of form
TO:timeValue, or timeValue not in range 0 through 32767.

Time slice or priority error

Timeslice or priority not in range 0 through 32767.

Undefined driver entry point

Specified driver entry point not relocated during system relocation
phase.

Undefined externals remain

Unknown command

Upper spool limit should be 1000 words less than SAM.

User table specification error

User table specification must be positive (and nonzero).

WARNING: LOCC set backward

The relocation pointer has been set backwards to overlay a
portion of previously created operating system.

WARNING: System common module not type 6

The generator expected the relocation of a type 6 module.

VCP Error Codes

There are three kinds of VCP errors: system bootup (pretest), input, and loader errors.

System Bootup Errors (Prior to Revision 4022)

Errors during pretest are displayed as patterns among the LEDs on the processor card (A600) or frontplane (A700). 1 indicates LED on. 0 indicates LED off. Where two codes are listed, they flash alternately. In this case, the right-hand six LEDs (indicated by ######) contain the pertinent information.

LEDs

Failure pattern

11111111   Microcoded self test failed or power supply problem. Power supply problem if PON (power-on signal) low. On A700, other lights indicate:

       11111111   Lower processor board bad
       01111111   Upper processor board bad
       00111111   Memory controller or front pane ROMs
       00011111   Floating point board bad

11111110   Microcode self test passed but first memory fetch failed. Probably memory controller board but could be processor.

11111100   Basic instruction test failed. Bad CPU.

11111000   Boot RAM failed. Bad memory controller.

Any failure after this point still allows the VCP to run if the VCP interface is good. The failure causes the pattern shown to flash once and then the VCP is entered displaying:

PTEST ERR XXXXXX XXXXXX

The first word is the main error code, and the second the flashing subcode.

Octal

11110000	360	TBG tick interrupt flags failed. Bad CPU
11100000	340	Main memory failure. The alternating
10######	2xx	code contains the bad 32k block. If ###### is 0, the problem could be in the parity circuitry or in the map RAMs, and could be caused by a bad memory controller.
11000000	300	I/O interface failure. Alternates with
	2xx	the 10###### interface select code or an error code. If ###### is greater than 17B, ###### is the failing select code.

######
values   **If ###### is less than 20B**

10000000	200	No I/O cards
10000001	201	More than one interface has break enable
10000010	202	Priority chain broken
10000011	203	Duplicate select code
10000100	204	Select code less than 20B
10000101	205	Terminal not connected for VCP
10000110	206	Unexpected time base generator interrupt
10000111	207	Unexpected memory protect interrupt
10001000	210	Unexpected UIT interrupt

PTEST ERR XXXXXX XXXXXX is the octal value of the lights. The first number is the main error value and the second number is the alternating code.

Example:

000300 000203 is I/O error (300) duplicate select code (203)   000340 100000 is memory error (340) in 32k block 0

System Bootup Errors (Revision 4022 and Later)

Pretest errors for Revision 4022 and later are shown in the following tables.

Input Errors

Entering an unrecognized command causes the VCP to respond:

!?
VCP>

Self-Test Pass Indications

Octal Code	LED Display	Definition
007⇌005	000001*1	VCP program is running; VCP console connected and waiting for user input.
002⇌000	000000*0	VCP program is running; VCP console is not connected, that is, VCP is waiting for the VCP console to respond to the first ENQ-ACK handshake.
001	00000001	A boot loader is running.
000	00000000	User software is running.
200	10000000	Loader error. Probably a checksum error; change media on loading device.
020	00010000	Running Diagnostic Design Language (DDL) Program.

1 = lit LED; 0 = unlit LED; * = flashing LED

Test 2 (phase 1) Error Codes

Error Code *		Meaning
Bank of 4 LEDs	Bank of 8 LEDs	
0000	11111110	Microcoded self-test (Test 1) passed but first memory fetch failed. Bad A990 board.
	11111100	
		Basic instruction test failed. Bad A990 board.
	11111000	
		Boot memory access failed. Bad EPROM or A990 board.
	11110000	
		TBG test failed. Bad A990 board.

* 1 = lit LED; 0 = unlit LED; X = don't care

Sample Display of Test 2 (phase 2) Error Code

Revision 4022 and later of VCP uses the following scheme to display the four-byte long error code on the 8 LEDs. Preceding each byte of information a "travelling" LED pattern is displayed to indicate which byte of information is going to be displayed. For example, the following LED sequence is displayed for the error code 302 201 023 021. See NO TAG for a definition of all the error codes.

Octal Code	LED Display	
302	○○○○○○○●	*One* travelling LED is displayed, then
	●●○○○○●○	*First* byte of error message is displayed
201	○○○○○○●●	*Two* travelling LEDs are displayed, then
	●○○○○○○●	*Second* byte of error message is displayed
023	○○○○○●●●	*Three* travelling LEDs are displayed, then
	○○○●○○●●	*Third* byte of error message is displayed
021	○○○○●●●●	*Four* travelling LEDs are displayed, then
	○○○●○○○●	*Fourth* byte of error message is displayed

Test 2 (phase 2) Error Codes

Octal Code *				Definition
Byte				
1	**2**	**3**	**4**	
360	360	xxx	xxx	TBG test failed. 3rd and 4th bytes (xxx xxx) indicate address in VCP where error was detected.
340	---	---	---	When LEDs are in this static pattern, memory test is running; this is not an error condition.
340	200	000	000	No memory found or missing frontplane connector.
340	2nn	xxx	xxx	Main memory failed. nn = 01b to 12b depending on where in VCP code the error was detected. 3rd and 4th bytes indicate 32-KB blk of failed memory.
341	000	377	377	ECC memory detected in A400/A600; ECC memory is not supported in an A400/A600. (This error should never occur on an A990.)
300	200	000	000	No I/O cards in the card cage.
300	201	0yy	0xx	More than one card has VCP enabled; yy and xx are the select codes of the enabled cards.
300	202	0yy	0xx	Broken I/O chain; yy is the number of I/O cards found by polling; xx is the number found by priority scan; the break will usually be found at slot xx+1.
300	203	000	0xx	Duplicate select codes; more than one card has the select code xx.
300	204	000	0xx	An I/O card at select code xx has a select code of less than 20b.
300	205	000	000	No I/O card has been enabled as VCP interface.
300	206	000	000	Unexpected TBG interrupt.
300	207	000	000	Unexpected Memory Protect interrupt.
300	210	000	000	Unexpected UIT interrupt.
300	211	000	000	Invalid ID number or select code for A400 OBIO. (This error should never occur on an A990.)
300	213	000	000	VCP Speed Sense failed.
300	220	xxx	xxx	} I/O card at sc 20 thru 77 failed self-test. xxx xxx is the address in VCP where error was detected.
300	277	xxx	xxx	

* Error codes for phase 2 of Test 2 are displayed on upper 8 LEDs as 4 bytes of information using scheme illustrated in previous figure.

VCP Loader Errors (In response to %B or %L)

LOADER ERROR 0   Unrecognizable load/bootstring.
LOADER ERROR 2   Select code less than 20B.
LOADER ERROR 3   No card with that select code.
LOADER ERROR 4   Product ID not found in EPROM (A990 only).

Cartridge Tape Loader Errors

LOADER ERROR 110   File forward error; status in B-Register.
LOADER ERROR 111   Checksum error.
LOADER ERROR 112   No data before EOF.
LOADER ERROR 113   Bad length word.
LOADER ERROR 120   Write error; status in B-Register.

PROM Card Loader Errors

LOADER ERROR 211   End of data on current card; bad file number.
LOADER ERROR 212   Bad format.
LOADER ERROR 213   System larger than 32K; must start on card boundary.
LOADER ERROR 214   Write not allowed to ROM.
LOADER ERROR 215   Timeout reading data.

DS/1000 Card Loader Errors (TO = Timeout)

LOADER ERROR 310   TO after CLC 0; is select code a DS card?
LOADER ERROR 311   Checksum error; is P file absolute binary?
LOADER ERROR 312   TO after download request.
LOADER ERROR 313   TO after file number.
LOADER ERROR 314   Bad transfer (cntrl request); status in B-Register.
LOADER ERROR 315   TO after buffer request.
LOADER ERROR 316   TO after count echo.
LOADER ERROR 317   TO waiting for data.
LOADER ERROR 320   TO during VCP mode request (%WDS).
LOADER ERROR 321   Remote node will not accept data (%WDS); status in B-Register.
LOADER ERROR 325   Record out of sequence; is central file type 6 file?

Disk Loader Errors (TO = Timeout)

LOADER ERROR 411	TO reading disk type or error in initialization process; check HP-IB address, card, cable; or, for SCSI, check condition after inquiry command.
LOADER ERROR 412	TO reading status; is device a disk?
LOADER ERROR 413	Status error; status in B-Register.
LOADER ERROR 414	TO during file mask.
LOADER ERROR 415	TO during seek. For SCSI, check condition after read/write, parity error/timeout during DMA transfer.
LOADER ERROR 416	TO during read/write command.
LOADER ERROR 417	TO during DMA of data.
LOADER ERROR 418	Parity error during DMA transfer.
LOADER ERROR 420	Parity error during DMA transfer.
LOADER ERROR 421	TO during FIFO flush.
LOADER ERROR 422	TO during DSJ command. *
LOADER ERROR 423	Bad DSJ return; returned value in B-Register. *
LOADER ERROR 460	Disk not identifiable; disk ID in B-Register.

Magnetic Tape Loader Errors (TO = Timeout)

LOADER ERROR 510	TO during initialization/read ID. For DAT tape, check condition after inquiry command.
LOADER ERROR 511	TO when issuing end/select unit.
LOADER ERROR 512	Magnetic tape offline.
LOADER ERROR 513	No write ring. For DAT tape, check condition after read/write, parity error/TO during DMA transfer.
LOADER ERROR 514	TO during End command.
LOADER ERROR 515	TO waiting for rewind completion.
LOADER ERROR 517	TO waiting for DMA transfer.
LOADER ERROR 520	Parity error during DMA transfer.
LOADER ERROR 521	TO doing a PHI flush.
LOADER ERROR 522	TO waiting for DSJ. *
LOADER ERROR 523	Bad DSJ response. *
LOADER ERROR 525	TO waiting for Mag Tape Not Busy.
LOADER ERROR 530	TO after issuing a command.
LOADER ERROR 531	Parallel Poll TO after issuing a command.
LOADER ERROR 535	Bad status after read/write command.
LOADER ERROR 550	No data transfer (read only).
LOADER ERROR 560	Not mag tape ID. For DAT tape, check condition after rewind command.

* DSJ = Device Specified Jump

HP 12022A Disk Interface Loader Errors (TO = Timeout)

LOADER ERROR 610	TO after SDH (sector driver head) for read/write.
LOADER ERROR 611	TO after cylinder high.
LOADER ERROR 612	TO after cylinder low.
LOADER ERROR 613	TO after sector.
LOADER ERROR 614	TO after sector count.
LOADER ERROR 615	TO after read/write command.
LOADER ERROR 616	TO after DMA read/write transfer.
LOADER ERROR 617	Parity error during transfer.
LOADER ERROR 620	Fixed disk not ready.
LOADER ERROR 630	TO after request status register.
LOADER ERROR 631	TO after read status register.
LOADER ERROR 632	TO after waiting for not busy.
LOADER ERROR 633	TO after request error register.
LOADER ERROR 634	TO after read error register.
LOADER ERROR 635	Status error: A-Register = status register; B-Register = error register.
LOADER ERROR 650	TO after SDH register for restore.
LOADER ERROR 651	TO after restore.
LOADER ERROR 660	Disk not defined.

Other

LOADER ERROR 1024/1025

Possible meanings:

1. Booting from CS/80 disk that has just been pushbutton restored from a CTD tape or booting diagnostics directly from the tape. The CTD tape may not have been certified/formatted before data was stored to it.

2. Booting from a CTD tape in ASAVE format.

3. Booting from the CS/80 disk was not successful. BOOTEX may be corrupt.

4. Faulty tape control board in the CS/80 disk.

5. Incorrect VCP file number in the bootstring.

The B-Register will give the following status:

0 = Ready for data; normal mode. Should not get 0 with this loader error.

1 = Status word should be read from drive. Indicates an error, or a new media could have been inserted.

2 = Powerup has occurred; no activity in the disk has occurred yet.

3 = Parity error on command.

VMA/EMA Error Codes

VMA/EMA errors that cause program abortion have the same format as the MP and DM error returns:

> VM xx

or

> EM xx

where xx is:

< 80 then xx is an FMP error number. FMP reports the error as a negative number and VMA/EMA reports the last two digits of that number.

It is possible that VMA/EMA will report different FMP errors by the same VMA/EMA error; for example, either FMP−006 (No such file) or FMP−206 (Directory read protected) is reported as VM06.

The VMxx error reported depends on the context. Examine the current status of the system to determine which of the several possible FMP errors may have been reported.

80   VMA/EMA system is corrupt.

81   Not a VMA/EMA program.

82   Requested page beyond maximum specified for VMA/EMA system, or the VMA disk is too small. x-reg = requested page number (in octal). y-reg = logical address to map in the requested page, abort address = address to instruction causing program to abort.

83   All pages locked.

84   File type not = 2 or record length not = 1024 words.

85   Scratch file cannot be purged.

86   Access to VMA system after the VMA file has been closed.

87   MSEG is too small.

88   Cannot respecify the VMA file.

89   Transfer too big for EMAIO or VMAIO.

VMA/EMA Error Codes (continued)

90   Shareable EMA size for program is larger than the
            shareable EMA area already allocated.

91   Program and shareable EMA area are assigned to
            same partition or program is assigned to a reserved
            partition in which program's shareable EMA area has
            already been allocated.

92   VMA or VMAIO is not available because %VEMA
            not genned.