MPE/iX Quick Reference Guide

HP 3000 MPE/iX Computer Systems

Edition 6

32650-90873
E1098

Printed in: U.S.A. October 1998

Notice

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

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for direct, indirect, special, incidental or consequential damages in connection with the furnishing or use of this material.

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

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c) (1,2).

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Contents

Contents

Tables

Tables

Preface

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s, not based on the PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.

The MPE/iX Quick Reference Guide offers a synopsis of the MPE/iX operating system and its major subsystems. Each chapter corresponds to a manual in the MPE/iX set. The table below lists the chapters in order and the corresponding MPE/iX manuals.

Chapter	Manual
Commands	*MPE/iX Commands Reference Manual Volumes 1 and 2*)
Utilities	*MPE/iX Utilities Manual*
Intrinsics	*MPE/iX Intrinsics Reference Manual*
FCOPY	*FCOPY Reference Manual*
SORT-MERGE	*SORT-MERGE/XL General User's Guide*
System Debug	*MPE/iX System Debug Reference Manual*
File System	*Using the 900 Series HP 3000: Fundamental Skills* *Using the 900 Series HP 3000: Advanced Skills*

Commands, intrinsics, and utility descriptions are in alphabetical order within the chapters. Each of the chapters shows syntax for commands and functions. Some chapters include examples; user input is underlined.

Use the Table of Contents to look up information within the sections.

1 Command Descriptions

Commands Syntax

These are abbreviated descriptions for the commands for MPE/iX.

ABORT

Aborts the current program or operation.

```
ABORT
ABORT
```

ABORTIO/ =ABORTIO

Aborts one pending I/O request for a device.

```
ABORTIO ldev

=ABORTIO ldev

ABORTIO 53
ABORTIO 53
```

ABORTJOB/ =ABORTJOB

Aborts a job or session.

```
ABORT JOB { #Jnnn
            #Snnn
            [jobname,] user.acct }

=ABORTJOB { #Jnnn
            #Snnn
            [jobname,] user.acct }

ABORTJOB #S139
```

ACCEPT

Permits a designated device to accept jobs/sessions and/or data.

```
ACCEPT [ JOBS
         DATA ] ,ldev

ACCEPT 19
```

ALLOCATE

Loads a compatibility mode program or procedure into virtual memory.

```
ALLOCATE [ PROCEDURE,
           PROGRAM,    ] name

ALLOCATE PROCEDURE, PROC1
```

ALLOW

Grants a user access to a specific operator command.

```
ALLOW FILE= formaldesignator[;SHOW]
```

or

```
ALLOW { @.@
        user.@
        @.acct
        user.acct } ;COMMANDS=command[,command, ...]

ALLOW FILE=ALLOWTMP;SHOW
ALLOW USER.TECH;COMMANDS=REPLY,ABORTIO
```

or

```
ALLOW
>MGR.MANUAL;COMMANDS=BREAKJOB
>EXIT
```

ALTACCT

Changes the attributes of an existing account. See Chapter 7 for a listing of account capabilities and defaults.

```
ALTACCT acctname

[;PASS=[password]][;FILES=[filespace]][;CPU=[cpu]]
[;CONNECT=[connect]]

[;CAP=[capabilitylist]][;ACCESS=[(fileaccess)]]
[;MAXPRI=[subqueuename]]

[;LOCATTR=[localattribute]][;ONVS=volumesetname]
ALTACCT AC2;PASS=GLOBALX;FILES=50000
```

ALTGROUP

Changes one or more attributes of a group.

```
ALTGROUP groupname[.acctname]  [;PASS=[password]]
[;CAP=[capabilitylist]]

[;FILES=[filespace sectors]] [;CPU=[cpu seconds]]
[;CONNECT=[connect minutes]]

[;ACCESS=[(fileaccess)]] [;ONVS=volumesetname]
[;HOMEVS=volumesetname]

ALTGROUP GROUPX;PASS=PASS2

ALTGROUP LEILA;ONVS=TIME_LORD;FILES=10000
ALTGROUP LEILA;HOMEVS=DICHONDRITE
```

ALTJOB

Alters the attributes of waiting or scheduled jobs.

```
ALTJOB [JOB=] {#Jnnn
               #Snnn}  [;INPRI=inputpriority]
[;OUTDEV={ldev
          devclass}][;JOBQ=jobqueuename] [;HIPRI]
ALTJOB #J1;INPRI=10;OUTDEV=LP
```

ALTLOG

Alters the attributes of an existing user logging identifier.

```
ALTLOG logid[;LOG=logfile {,DISC
                           ,TAPE}][;PASS=password]
[{;AUTO
  ;NOAUTO}]
ALTLOG KIM;LOG=C,DISC
```

ALTPROC

Changes the priority for the specified process(es) *if* you have OP or SM capabilty. **Native Mode**

```
ALTPROC [ [;PIN=] {pinspec  }]
                   {(pinspec[,pinspec]...)}
        [ [;JOB=] {jobspec }]
                   {(jobspec[,jobspec]...)}
        { [;PRI=] pri
          [;WG=]{workgrp
                 NATURAL_wg }}
        [;TREE | ;NOTREE]
        [;USER | ;ANYUSER]
        [;SYSTEM]

ALTPROC 42;PRI=CM
ALTPROC O;PRI=DS
ALTPROC job=mgr.payroll; PRI=155
ALTPROC #p133;TREE;PRI=CS
ALTPROC (150,#p247,211);PRI=ES
ALTPROC job=@j; PRI=CS;ANYUSER
ALTPROC job=@j;PRI=CS; USER
```

ALTSEC

Changes the access permissions by altering the access control definition (ACD). Access permissions may be changed for a

- file
- hierarchical directory
- device
- device class

File access masks can also be changed with this command (only files have access masks). The file status change time stamp is updated by ALTSEC.

NOTE	The ALTSEC command does not change access permissions for MPE groups, accounts, or the root directory.

Syntax

```
ALTSEC objectname [,{FILENAME
                     LDEV
                     DEVCLASS} ]
[;[ACCESS=](fileaccess[;[fileaccess][;...]])]
[ {;NEWACD=
   ;REPACD=
   ;ADDPAIR=
   ;REPPAIR=} {(acdpair [;acdpair][;...] )
                ^filereference } ]
[;DELPAIR= {(userspec [;userspec][;...])
             ^filereference } ]
[;COPYACD= objectname {,FILENAME
                       ,LDEV} ]  [;DELACD][;MASK]
```

ALTSPOOLFILE

Alters the characteristics of an output spoolfile.

```
ALTSPOOLFILE {#Onnn
              ldev1} {;PRI=outputpriority
                      ;COPIES=numcopies
                      ;DEV={ldev2
                            devclass}
                      ;DEFER              } |;...|
ALTSPOOLFILE #O86;DEFER
ALTSPOOLFILE 6;DEFER
ALTSPOOLFILE #O123;PRI=3
```

ALTUSER

Changes the attributes currently defined for a user.

```
ALTUSER username[.acctname]
 [;PASS=[password]]
 [;CAP=[capabilitylist]]
 [;MAXPRI=[subqueuename]]
 [;LOCATTR=[localattribute]]
 [;HOME=[homegroupname]]
 [;UID=[uid]]
 [;USERPASS={REQ [,EXPIRED]}
            {OPT}]

ALTUSER JONES;CAP=IA,BA,SF,PH,DS,MR
ALTUSER JONES;PASS=JJ;MAXPRI=DS
```

ASSOCIATE

Gives a user operator control of a device class.

```
ASSOCIATE devclass
ASSOCIATE TAPE
```

BASIC

Interprets a compatibility mode BASIC/V program.

```
BASIC [commandfile][,[inputfile][,listfile]]
BASIC MYCOMDS,MYDATA,MYLIST
```

BASICGO

Compiles, prepares, and executes a compatibility mode BASIC/V program.

```
BASICGO [commandfile][,listfile]
BASICGO
$CONTROL USLINIT
$COMPILE MYPROG
$EXIT
```

BASICOMP

Compiles a compatibility mode BASIC/V program.

```
BASICOMP [commandfile][,[uslfile][,listfile]]
BUILD OBJECT;CODE=USL
BASICOMP, OBJECT
$CONTROL USLINIT
$COMPILE MYPROG
$EXIT
```

BASICPREP

Compiles and prepares a compatibility mode BASIC/V program.

```
BASICPREP [commandfile][,[progfile][,listfile]]
BASICPREP,MYCOMDS
```

BBASIC

Starts execution of the HP Business BASIC/V interpreter in compatibility mode.

```
BBASIC [commandfile][,[inputfile][,listfile]]
BBASIC
```

BBASICGO

Compiles, prepares, and executes an HP Business BASIC/V program in compatibility mode.

```
BBASICGO infile[,listfile]
BBASICGO MYPROG,LISTFL
```

BBASICOMP

Compiles an HP Business BASIC/V program in compatibility mode.

```
BBASICOMP infile[,[uslfile][,listfile]]
BBASICOMP MYPROG,OBJECT
```

BBASICPREP

Compiles and prepares a HP Business BASIC/V program in compatibility mode.

```
BBASICPREP infile[,[progfile][,listfile] ]
BBASICPREP MYCOMDS,MYPROG
```

BBXL

Initiates execution of the HP Business BASIC/XL interpreter.

```
BBXL [commandfile][,[inputfile][,[listfile]]] [;XL=xllist]
BBXL
```

BBXLCOMP

Compiles an HP Business BASIC/XL program.

```
BBXLCOMP textfile[,[objectfile][,listfile]]
BBXLCOMP MYPROG,OBJECT
```

BBXLGO

Compiles, links, and executes an HP Business BASIC/XL program.

```
BBXLGO textfile[,[listfile]] [;XL=xllist]
BBXLGO MYPROG,LISTFL
```

BBXLLK

Compiles and links an HP Business BASIC/XL program.

```
BBXLLK textfile[,[progfile][,listfile]]
BBXLLK MYSCR,MYPROG
```

BREAKJOB

Suspends an executing job.

```
BREAKJOB #Jnnn
BREAKJOB #J68
```

BUILD

Creates and immediately allocates a new empty file on disk.

```
BUILD filereference
[;REC=[ [recsize][,[blockfactor][,[F
                                    U
                                    V
                                    B][,BINARY
                                       ,ASCII ]]]]]
[;CCTL
 ;NOCCTL] [;TEMP] [;DEV=  [ dsdevice#
                            dsdevice#device
                            device ] ]
[;CODE=filecode] [;DISC=[ [numrec][,[numextents][,initialloc]]]]
[;RIO
 ;NORIO] [;MSG
          ;CIR
          ;STD
          ;KSAMXL
          ;SPOOL] [;ULABEL=numlabels] [;KEY={^filereference
                                             keyinfo} ]
[;FIRSTREC=recnum][;REUSE
                  ;NOREUSE]
```

Where:

```
;KEY=(keytype,keylocation,keysize [,DUP
                                  ,RDUP];
.
.
.

keytype,keylocation,keysize [,DUP
                            ,RDUP])
BUILD WORKFILE;REC=-80,3,F,ASCII;DISC=2000,10,2
BUILD VFILE;DISC=500,10,1;REC=-80;DEV=VCLASS1
BUILD NEWDATA;DISC=3000,1,1;CODE=LOG
```

BYE

Ends an interactive session.

```
BYE
BYE
```

CALC

Evaluates an expression.

```
CALC expression
CALC 5*10-7
```

CCXL

Compiles an HP C/iX program.

```
CCXL [textfile][,[objectfile][,[listfile]]] [;INFO=quotedstring]
CCXL
```

CCXLGO

Compiles, links, and executes an HP C/iX program.

```
CCXLGO [textfile][,[listfile]][;INFO=quotedstring]
CCXLGO SOURCE,LISTFILE
```

CCXLLK

Compiles and links an HP C/iX program.

```
CCXLLK [textfile][,[ [progfile]][,[listfile]]] [;INFO=quotedstring]
CCXLLK SOURCE,PROG
```

CHANGELOG

Changes the user logging file without stopping or interrupting the logging process.

```
CHANGELOG logid [;DEV=device]
CHANGELOG KATHY
```

CHDIR

Changes the process' current working directory (CWD).

Syntax

```
CHDIR [ [DIR=]dir_name] [;SHOW | NOSHOW]
CHDIR /MYACCT/MYGRP/dir1|
```

CHGROUP

Switches you from the current group to any other group within the logon account to which you are allowed access.

```
CHGROUP [ [groupname] [/grouppass] ]
CHGROUP GORODA
```

COB74XL

Compiles an HP COBOL II/XL program using the 1974 ANSI standard entry point and creates an object file.

```
COB74XL [textfile] [,[objectfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring] [;WKSP=workspacename] [;XDB=xdbfilename]
COB74XL SOURCE,OBJECT,LISTFL
```

COB74XLG

Compiles, links, and executes an HP COBOL II/XL program using the ANSI 1974 standard entry point.

```
COB74XLG  [textfile][,[listfile][,[masterfile][,newfile]]]
[;INFO=quotedstring] [;WKSP=workspacename] [;XDB=xdbfilename]
COB74XLG TEXTFL,LISTFL
```

COB74XLK

Compiles and links an HP COBOL II/XL program using the 1974 ANSI standard entry point.

```
COB74XLK [textfile] [,[progfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring] [;WKSP=workspacename] [;XDB=xdbfilename]
COB74XLK SFILE,MYPROG
```

COB85XL

Compiles an HP COBOL II/XL program using the 1985 ANSI standard entry point and creates an object file.

```
COB85XL [textfile] [,[objectfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring] [;WKSP=workspacename] [;XDB=xdbfilename]
COB85XL SOURCE,OBJECT,LISTFL
```

COB85XLG

Compiles, links, and executes an HP COBOL II/XL program using the ANSI 1985 standard entry point.

```
COB85XLG [textfile][,[listfile][,[masterfile]
                                [,newfile]]]
[;INFO=quotedstring] [;WKSP=workspacename] [;XDB=xdbfilename]
COB85XLG TEXTFL,LISTFL
```

COB85XLK

Compiles and links an HP COBOL II/XL program using the 1985 ANSI standard entry point.

```
COB85XLK [textfile] [,[progfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring] [;WKSP=workspacename] [;XDB=xdbfilename]
COB85XLK SFILE,MYPROG
```

COBOLII

Compiles a compatibility mode COBOLII program on the COBOL 74 compiler.

```
COBOLII [textfile][,[uslfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring] [;WKSP=workspacename]
BUILD OBJECT;CODE=USL
COBOLII SOURCE,OBJECT,LISTFL
```

COBOLIIGO

Compiles, prepares, and executes a compatibility mode COBOLII program on the COBOL 74 compiler.

```
COBOLIIGO [textfile][,[listfile][,[masterfile][,newfile]]]
[;INFO=quotedstring][;WKSP=workspacename]
COBOLIIGO TEXTFL,LISTFL
```

COBOLIIPREP

Compiles and prepares a compatibility mode COBOLII program on the COBOL 74 compiler.

```
COBOLIIPREP [textfile]
[,[progfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring][;WKSP=workspacename]
COBOLIIPREP SFILE,MYPROG
```

COMMENT

Inserts a comment into the command stream.

```
COMMENT [text]
!JOB USER.TECHPUBS
!COMMENT  THIS IS A SAMPLE JOB
!FORTGO MYPROG
!EOJ
```

CONSOLE

Changes the system console from its current device to another job-accepting terminal.

```
CONSOLE [ldev]
CONSOLE 31
```

CONTINUE

Overrides any job error which may occur in the next command so that the job or user command (command file or UDC) continues executing.

```
CONTINUE
!JOB USER.PUBS
!CONTINUE
!RUN MYPROG
.
.
.
!EOJ
```

COPY

Copies one file to another by creating a new file or by overwriting an existing file. The COPY command can be used to copy files to and from HFS directories. Also, users with SM capabilities are able to copy files to MPE accounts outside of their current logon account.

Syntax

```
COPY [FROM=]sourcefile[[;TO=]targetfile][ ;ASK | YES | NO ]

COPY ABCD, EFG
COPY ABCD,.newgroup
COPY ABCD.grp
COPY ABCD.grp,.mygroup

COPY myfile.pub.sys, ./MyFile
COPY ./File1, myfile.pub
COPY ./dir0/file1, ./dir1/file2
```

DATA

Enters data into the system from a device file. (Cannot be used to enter data from $STDIN.)

```
DATA[jsname,]username[/userpass].acctname[/acctpass][;filename]
DATA SESSB,BROWN.ACCT1
.
.
.
:EOD
```

DEALLOCATE

Deallocates a program or procedure previously loaded into memory with the ALLOCATE command.

```
DEALLOCATE [PROGRAM
            PROCEDURE],name
DEALLOCATE PROGEX
```

DEBUG

Instructs MPE/iX to enter the system debugger.

```
DEBUG [commands]
DEBUG TRACE;C
```

DELETESPOOLFILE

Deletes a spoolfile from disk.

```
DELETESPOOLFILE {#Onnn
                 #Innn
                 ldev }
DELETESPOOLFILE 6
```

DELETEVAR

Deletes one or more MPE/iX session variables.

```
DELETEVAR varname[,varname]...[,varname]
DELETEVAR  firstvariable,secondvariable
DELETEVAR JOBNUM, SESSNUM
```

DISALLOW

Prohibits access to a specific operator command.

```
DISALLOW FILE=formaldesignator[;SHOW]
DISALLOW {@.@
          user.@
          @.acct
          user.acct};COMMANDS=command[,command,...]
DISALLOW USER.TECH;COMMANDS=REPLY,ABORTIO
```

DISASSOCIATE

Removes control of a device class from the user.

```
DISASSOCIATE devclass
DISASSOCIATE TAPE
```

DISCRPS

Enables or disables the rotational position sensing (RPS) feature on a specified logical device.

```
DISCRPS ldev {,ENABLE [{,value,value}]
             ,DISABLE}
DISCRPS 1,ENABLE
```

DISKUSE

Displays disk space usage, in sectors, for one or more directories or a directory tree.

Syntax

```
DISKUSE [ [DIR=]dir_name][; TREE | NOTREE | USENAME ]
DISKUSE mydir.group.acct
```

DISMOUNT

Causes a volume set that was explicitly reserved by the user (with a MOUNT or VSRESERVE command) to be released. The equivalent MPE/iX command is VSRELEASE.

```
DISMOUNT [{*
           (blank)
           volumesetname}][.groupname[.acctname]]
DISMOUNT MYSET.B.C
```

DO

Allows the user to reexecute any command still retained in the command line history stack.

```
DO [CMD=cmdid][;EDIT=editstring]
DO 10
```

DOWN

Removes a device from normal system use. This command does not apply to disks.

```
DOWN ldev
DOWN 7
```

DOWNLOAD

Downloads format information to a line printer.

```
DOWNLOAD ldev [,filename
              ,MARGIN=nn] [,...]
DOWNLOAD 11,VFCPAY
```

DSTAT

Displays the current status of the disk drives on the system.

```
DSTAT [ldev
       ALL]
DSTAT ALL
```

ECHO

Displays a message on the standard list device.

```
ECHO [message]
SETVAR a, 'hi there'
ECHO !a, Cathy
```

EDITOR

Starts the EDIT/3000 subsystem.

```
EDITOR [listfile]
FILE LISTFILE;DEV=LP
EDITOR *LISTFILE
```

ELSE

Provides an alternate execution sequence for an IF statement in a jobfile or user command within an IF statement.

```
ELSE
!CONTINUE
!PASXL MYPROG,MYUSL
!IF JCW>=FATAL THEN
!   TELL USER.TECHPUBS;COMPILE FAILED
!ELSEIF JCW ≥ WARN THEN
!   TELL USER.TECHPUBS;COMPILE COMPLETED WITH WARNINGS
!ELSE
!   TELL USER.TECHPUBS;COMPILE COMPLETED WITH NO WARNINGS
!ENDIF
```

ELSEIF

Provides an alternate execution sequence for an IF statement.

```
ELSEIF expression [THEN]
IF EXPN1 THEN
   ...
ELSEIF EXPN2 THEN
   ...
ELSEIF EXPN3
   ...
ELSE
   ...
ENDIF
```

ENDIF

Terminates an IF block.

```
ENDIF
```

```
IF
.
.
.
ENDIF
```

ENDWHILE

Terminates a WHILE block.

```
ENDWHILE
WHILE logical_expression
    .
    .
    .
ENDWHILE
```

EOD

Denotes end-of-data on input stream from a jobfile (from an input other than $STDIN). It also terminates data initialized by the DATA command. The colon (:) is a required part of this command.

```
:EOD
DATA SESS1,BLACK.ACCTSP
     ... data ...
:EOD
```

EOJ

Ends a batch job.

```
EOJ

!JOB USER.PUBS
!RUN MYPROG1
!RUN MYPROG2
!EOJ
```

ERRCLEAR

Zeros out all HP predefined error-related variables.

```
ERRCLEAR

ERRCLEAR
```

ERRDUMP

Allows a user to dump either the process or system error stack to a specified depth.

```
ERRDUMP [errorstackdepth][;SYS]

ERRDUMP 1;SYS
```

ESCAPE

Allows the CI programmer to simulate all aspects of CI error handling. Control leaves all user commands and returns to the CI (unless a CONTINUE is in effect).

```
ESCAPE [ [CIERR=] errnum]

cmd1
CONTINUE
udc1
            ucmdA
            ucmdB
            ESCAPE
cmd2
```

EXIT

Terminates the command interpreter.

```
EXIT
```

```
EXIT
```

FCOPY

Runs the FCOPY subsystem.

```
FCOPY [fcopycommand]
FCOPY FROM=UDC.TECHPUBS;TO=TEMP;NEW
```

FILE

Declares the file attributes to be used when a file is opened. This declaration, informally known as a file equation, may be used to override programmatic or system default file specifications. With the addition of shared parameters from the NS3000/XL AdvanceNet subsystem, the declaration may specify a formal file designator that may be used to access a remote file or device in a subsequent command or intrinsic. NS3000/XL AdvanceNet is not part of the 900 Series HP 3000 Computer System Fundamental Operating System and must be purchased separately.

```
FILE formaldesignator =[*formaldesignator
                       $NULL
                       $NEWPASS
                       $OLDPASS
                       $STDIN
                       $STDINX
                       $STDLIST
                       filereference]
[:nodespec
 ,filedomain] [;DEV=[  [envname]#][device][,[outpri][,numcopies]]]
[;VTERM] [;ENV=envfile[:nodespec]][;option][;access][;DEL
                                                     ;TEMP
                                                     ;SAVE
                                                     ;SPSAVE]
```

Syntax for Option:

```
[;REC=[recsize][,[blockfactor][,[F
                                U
                                V]]
               B[,BINARY

               ,ASCII ]]]
[;DEN=[density]][;DISC=[numrec][,[numextents][,initialloc]]]
[;CODE=filecode]
[;RIO
 ;NORIO] [;STD
          ;MSG
          ;CIR
          ;KSAMXL
          ;SPOOL] [;ULABEL=numlabels]
[;KEY={^filereference
       keyinfo}] [;FIRSTREC=recnum] [;REUSE
                                     ;NOREUSE]
```

Syntax for *keyinfo*:

```
;KEY=(keytype,keylocation,keysize [,DUP
                                  ,RDUP]; [vellip]keytype,keylocation,keysize
[,DUP
 ,RDUP])
```

Syntax for Access:

```
[;NOCCTL  [;NOMULTI   [;NOMR    [;WAIT      [;ACC= [IN
 ;CCTL]    ;MULTI     ;MR  ]    ;NOWAIT]           OUT
           ;GMULTI]                                UPDATE
                                                   OUTKEEP
                                                   APPEND
                                                   INOUT] ]
```

```
[;BUF=[numbuffers]   [;LOCK      [;COPY
 ;NOBUF]                                  ;NOLOCK ]   ;NOCOPY ] [;FORMS=formsmsg]
```

```
[;EXC
 ;SHR
 ;EAR
 ;SEMI]

 [;NOLABEL
 ;LABEL=[ [volid][,[IBM
                   ANS][,[expdate][,seq] ] ] ] ][;FORMID=formid]
[;PRIVATE]
```

```
FILE SOURCE=INX
FILE DEST=OUTX
RUN MYPROG
FILE DEST=FILEX,NEW;REC=64,2,F,ASCII;DISC=800,10,2;SAVE
RUN MYPROG
FILE SOURCE=TAPE1,OLD;DEV=TAPE;REC=-80
FTNXL *SOURCE
FILE FTNTEXT=*SOURCE
FILE X=./my_file;SAVE
PURGE *X
```

FINDDIR (UDC)

The FINDDIR UDC executes the LISTFILE command to search for a directory.

NOTE System-defined UDCs are not automatically available. Your System Manager must use the SETCATALOG command to make these UDCs available for your use. For example:

```
SETCATALOG HPPXUDC.PUB.SYS
```

Syntax

```
FINDDIR [ [DIR=]dir_name][ [START=]start_dir]
```

Refer to the LISTFILE command later in this chapter for examples.

FINDFILE (UDC)

The FINDFILE UDC executes the LISTFILE command to search for a file.

NOTE System-defined UDCs are not automatically available. Your System Manager must use the SETCATALOG command to make these UDCs available for your use. For example:

```
SETCATALOG HPPXUDC.PUB.SYS
```

Syntax

```
FINDFILE [FILE=]filename[ [START=]start_dir]
```

FORMSALIGN

Configures one spooled printer or a group of spooled printers related by device class, to conditionally enter into a forms message dialog with its operator when the current spoolfile includes a forms message.

```
FORMSALIGN [ DEV= ] { ldev
                      devclass
                      devname }
[;[DIALOG=] { { EACHCHANGE
                EACHFILE
                EACHCOPY } [, { FORMIDOVERRIDE
                                NOFORMIDOVERRIDE } ] }    ]
[;SHOW]
FORMSALIGN LP;SHOW
```

FORTGO

Compiles, prepares, and executes a compatibility mode FORTRAN 66/V program.

```
FORTGO [textfile][,[listfile][,[masterfile][,[newfile] ] ] ]
[;INFO=quotedstring]
FORTGO SOURCE,LISTFL
```

FORTPREP

Compiles and prepares a compatibility mode FORTRAN 66/V program.

```
FORTPREP [textfile][,[progfile][,[listfile][,[masterfile][,[newfile]]]]]
[;INFO=quotedstring]
FORTPREP TEXTX,PROGX,LISTX
```

FORTRAN

Compiles a compatibility mode FORTRAN 66/V program.

```
FORTRAN [textfile][,[uslfile][,[listfile][,[masterfile][,[newfile] ] ] ] ]
[;INFO=quotedstring]
FORTRAN MYSOURCE,MYUSL,MYLIST;INFO= "$CONTROL BOUNDS"
```

FREERIN

Releases a global resource identification number (RIN).

```
FREERIN rin
FREERIN 1
```

FTN

Compiles a compatibility mode FORTRAN 77/V program.

```
FTN [textfile][,[uslfile][,[listfile]]][;INFO=quotedstring]
BUILD FORTOBJ;CODE=USL
FTN FORTSRC,FORTOBJ,LISTFILE
```

FTNGO

Compiles, prepares, and executes a compatibility mode HP FORTRAN 77/V program.

```
FTNGO [textfile][,listfile][;INFO=quotedstring]
FTNGO FORTSRC,LISTFILE
```

FTNPREP

Compiles and prepares a compatibility mode HP FORTRAN 77/V program.

```
FTNPREP [textfile],[progfile][,listfile][;INFO=quotedstring]
FTNPREP FORTSRC,FORTPROG
```

FTNXL

Compiles an HP FORTRAN 77/iX program.

```
FTNXL [textfile][,[objectfile][,[listfile]]][;INFO=quotedstring]
FTNXL FORTSRC,FORTOBJ,LISTFILE
```

FTNXLGO

Compiles, links, and executes an HP FORTRAN 77/iX program.

```
FTNXLGO [textfile][,[listfile]][;INFO=quotedstring]
FTNXLGO FORTSRC,LISTFILE
```

FTNXLLK

Compiles and links an HP FORTRAN 77/iX program.

```
FTNXLLK [textfile][,[progfile][,[listfile]]][;INFO=quotedstring]
FTNXLLK FORTSRC,FORTPROG
```

GETLOG

Establishes a logging identifier on the system.

```
GETLOG logid;LOG=logfile [,DISC
                         ,TAPE] [;PASS=password][;AUTO
                                                ;NOAUTO]
GETLOG FINANCE;LOG=A,DISC
```

GETRIN

Acquires a global resource identification number (RIN) and assigns a password to it.

```
GETRIN rinpassword
GETRIN MYRIN
```

HEADOFF

Stops header/trailer output to a device.

```
HEADOFF ldev
HEADOFF 6
```

HEADON

Resumes header/trailer output to a device.

```
HEADON ldev
HEADON 6
```

HELLO

Initiates an interactive session.

```
HELLO [session,]user[/userpass].acct[/acctpass][,group[/grouppass]]
[;TERM={termtype
        termname}][;TIME=cpusecs][;PRI={BS
                                        CS
                                        DS
                                        ES}]
[;INPRI=inputpriority
 ;HIPRI][;INFO=ciinfo] [;PARM=ciparm]
HELLO USER.TECHPUBS
```

HELP

Accesses the HELP subsystem.

Direct access:

```
HELP  [{udcfilename
       commandname[ keyword][,ALL]
       commandfilename
       programfilename
       SUMMARY
       CLASS
       HELPSTUDY
       FUNCTIONS
       EXPRESTIONS
       VARIABLES
       OPERATORS }]
```

Interactive (subsystem) access:

```
commandname {space or comma} keyword [,ALL]

   HELPMENU
   SUMMARY
   CLASS
   HELP
   HELPSTUDY

HELP ABORT
HELP LINKALL.TEST.UI
```

IF

Used to control the execution sequence of a job, UDC, or command file.

```
IF  expression [THEN]
!PASXL MYPROG,MYUSL
!IF JCW>=FATAL THEN
!   TELL USER.TECHPUBS;COMPILE FAILED
!ELSE
!   TELL USER.TECHPUBS;COMPILE COMPLETED
!ENDIF
```

INPUT

Permits the user to assign a string to any variable. All numeric input is treated as a string. See TYPEOF function in appendix B of the *MPE/iX Commands Reference Manual*, and SETVAR command in the same manual.

```
INPUT [NAME=]varname[;PROMPT=prompt][;WAIT=seconds][READCNT=chars]
INPUT Response; "Enter YES or NO>"
INPUT Response; "Press any key to continue"
```

JOB

Defines a job to be activated with the STREAM command to run in batch mode.

```
JOB [jobname,]username[/userpass].acctname[/acctpass]
[,groupname[/grouppass] ]
[;TIME=cpusecs][;PRI={BS
                     CS
                     DS
                     ES}]
[;INPRI=inputpriority
 ;HIPRI]
[;RESTART] [;OUTCLASS=[ [device][,[outputpriority][,numcopies]]]]
[;TERM={termtype}] [;PRIVATE] [;SPSAVE][JOBQ=queuename]
RUN EDITOR.PUB.SYS
/ADD
       1 !JOB WXYZ,WRITER.TEC
       2 !EDITOR
       3 TEXT ABC
       4 LIST ALL,OFFLINE
       5 EXIT
       6 !EOJ
         //
/KEEP MYJOB
/EXIT
:
STREAM MYJOB
STREAM
JOB USER.TECHPUBS;OUTCLASS=12
```

JOBFENCE

Defines the minimum input priority that a job or session must have in order to execute.

```
JOBFENCE priorityfence
JOBFENCE 8
```

JOBPRI

Sets or changes the default execution priority for batch jobs and sets a maximum execution priority for batch jobs.

```
JOBPRI [maxsubqueue][,defaultsubqueue]
JOBPRI 0
```

JOBSECURITY

Designates what level of user may request resources and control the execution of jobs.

```
JOBSECURITY {HIGH
             LOW }
            [;PASSEXEMPT={none} {user},{xaccess},{max}]
JOBSECURITY LOW
```

LDISMOUNT

Negates a previously issued LMOUNT or VSRESERVE command. This informs the system that the volume set is no longer reserved system-wide. The equivalent native mode command is VSRELEASESYS.

```
LDISMOUNT [*
          volumesetname][,groupname[.acctname]]
LDISMOUNT DATABASE.PAYROLL.ACCTNG
```

LIMIT

Limits the number of concurrently running jobs/sessions.

```
LIMIT [[+ | -]numberjobs]
      [[+ | -],numbersessions ][;JOBQ=qname]

LIMIT 2,15
```

LINK

Creates an executable program file by merging the relocatable object modules from all the files in its FROM= parameter.

```
LINK [FROM=file[,file[,...]][;TO=destfile]]
[;RL=rlfile[,rlfile[,...]]][;XL=xlfile[,xlfile[,...]]]
[;CAP=caplist] [;NMSTACK=nmstacksize][;NMHEAP=nmheapsize]
[;UNSAT=unsatname]
[;PARMCHECK=checklevel] [;ENTRY=entryname][;NODEBUG][;MAP]
[;SHOW]
LINK FROM=OBJCODE;TO=EXECPROG;NMSTACK=50000;MAP;SHOW
```

LISTACCT

Displays information about one or more accounts.

Syntax

```
LISTACCT [acctset][,listfile][;PASS]
[;FORMAT={SUMMARY|BRIEF|DETAIL}]
LISTACCT HPXLII;PASS
```

LISTDIR (UDC)

The LISTDIR UDC executes the LISTFILE command to list all files that are directories.

NOTE System-defined UDCs are not automatically available. Your System Manager must use the SETCATALOG command to make these UDCs available for your use. For example,

```
SETCATALOG HPPXUDC.PUB.SYS
```

Syntax

```
LISTDIR [ [DIR=]dir_name][ [FORMAT=]format_opt]
```

LISTEQ

Displays all active file equations for a job or session.

```
LISTEQ [listfile]
```

```
LISTEQ

FILE EQUATIONS

FILE TAPE1;DEV=ATAPE
FILE PP;ENV=LP2.ENV.OSE;DEV=EPOC
FILE MYFILE,NEW;REC=-80,3,F,ASCII;DISC=5000;SAVE
FILE POSIX=./mydir/myfile1
```

LISTF

Displays information about one or more permanent files.

```
LISTF [fileset][,listlevel][;listfile]
LISTF
```

LISTFILE

Lists file information.

```
LISTFILE [{fileset ((fileset [,fileset ])}][ [;FORMAT=] format_opt]
[ [;SELEQ=] select_eq]
[ [;NAME=] pattern ]
[;PASS][;PERM
         ;TEMP
         ;PERMTEMP]
[ ;USENAME
  ;TREE
  ;NOTREE ]
```

Selection equations, *enclosed in square brackets*, have the following format:

```
[FTYPE = KSAMXL | SPOOL]   [OBJECT = ACCT | GROUP | FILE | DIR]

[ACCESS = INUSE | OPEN | LOCKED | EXCLUSIVE]

[CODE = filecodenumber |PRIV | filecode mnemonic]
```

NOTE Selection equations must be surrounded by square brackets.

```
LISTFILE KSAMFMT, 7
LISTFILE ,DISC
LISTFILE [a-f]#[g-z@],3;SELEQ=[FTYPE=SPOOL]
```

LISTFTEMP

Displays information about one or more temporary files.

```
LISTFTEMP [fileset][,listlevel][;listfile]
```

```
LISTFTEMP
```

LISTGROUP

Displays information for one or more groups.

```
LISTGROUP [groupset][,listfile][;PASS][;FORMAT={SUMMARY|BRIEF|}]

LISTGROUP DEVELOP;PASS;FORMAT=SUMMARY
LISTGROUP @.@;FORMAT=BRIEF
```

LISJOBQ

Displays all job queues on system.

LISTLOG

Lists currently active logging identifiers on the system and whether automatic log file changing has been enabled.

```
LISTLOG [logid[;PASS]]
LISTLOG
```

LISTREDO

Displays the contents of the command line history stack. You may specify the format in which the listing will appear, and whether it will appear on $STDLIST or in a file.

```
LISTREDO [START=m][;END=n][;OUT=outfile][;ABS
                                        ;REL
                                        ;UNN]
LISTREDO -10,-2;OUT=*LIST;UNN
```

LISTSPF

```
LISTSPF[  [IDNAME=]  { spoolid
                       (spoolid  [,spoolid]...)}]
[  [;SELEQ=] { select-eq
               ^indirect_file  }]    [;DETAIL
                                        ;STATUS]
```

Where the select equation, *enclosed in square brackets*, has the following format:

```
select-eq  : : = [equation]

equation  : : = {parm  {>
                        >=
                        <
                        <= <>
                        =}  value
                NOT (equation)
                (equation)  {AND
                             OR}  (equation)  }
LISTSPF O@;SELEQ=[(PRI>2)AND(DATE<09/30/89)];DETAIL
```

LISTUSER

Displays information for one or more users.

```
LISTUSER [userset][,listfile][;PASS]
[;FORMAT={SUMMARY|BRIEF|DETAIL}]
LISTUSER PETE;PASS
LISTUSER PETE;PASS;FORMAT=SUMMARY
LISTUSER @;FORMAT=BRIEF
LISTUSER PETE;FORMAT=DETAIL
```

LMOUNT

Requests a logical reservation of a volume set. This informs the system that the volume set is to be reserved system-wide. The equivalent native mode command is VSRESERVESYS.

```
LMOUNT [{*
         (blank)
         volumesetname}] [.groupname[.acctname] ]
[;GEN=[genindex]]
LMOUNT DATABASE.PAYROLL.ACCTNG
VSRESERVESYS DATABASE.PAYROLL.ACCTNG
```

LOG

Starts, restarts, or stops user logging.

```
LOG logid {,RESTART
          ,START
          ,STOP}
LOG LOGPROCX,START
```

=LOGOFF

Aborts all executing jobs/sessions and prevents any further logon. You may optionally specify one job or one session that is to remain logged on.

```
=LOGOFF [#Snnn
         #Jnnn]
CTRL A
=LOGOFF
```

or

```
CTRL A
=LOGOFF #S2
```

=LOGON

Enables job/session processing following =LOGOFF.

```
=LOGON
CTRL A
=LOGON
```

MOUNT

Sends a request to the system to reserve a volume set (keep it online). The set must be on line in order to have the command take effect. The equivalent MPE/iX command is VSRESERVE.

```
MOUNT [{*
       (blank)
       vsname}][.groupname[.acctname] ][;GEN=[genindex]]
MOUNT MYSET;GEN=43
```

NEWACCT

Creates a new account with an associated account manager and PUB group.

```
NEWACCT acctname,mgrname[;PASS=[password]][;FILES=[filespace]]
[;CPU=[cpu]][;CONNECT=[connect]][;CAP=[capabilitylist]]
[;ACCESS=[fileaccess]]
```

```
[;MAXPRI=[subqueuename]][;LOCATTR=[localattribute]]
[;ONVS=volumesetname]
NEWACCT ACI,MNGR
NEWACCT DOCTOR,WHO;CAP=IA,BA,GL,AM,AL
NEWACCT DOCTOR,WHO;ONVS=MY_VOL
NEWACCT DOCTOR,WHO;UID=150;GID=120;CAP=IA,BA,SF,ND,GL,AM,AL
```

NEWDIR

Creates a directory.

Syntax

```
NEWDIR [DIR=]dir_name [;SHOW | NOSHOW]
NEWDIR /MYACCT/MYGRP/DIR1
NEWDIR dir1.mygroup.myacct
NEWDIR /myacct/jones/cmdf/john
```

NEWLINK

This command creates a link to a file, group, account, or directory.

```
NEWLINK [ LINK=] linkname[;TO=]   sourceobject[ {;SYMBOLIC} ]
:NEWLINK LINK=PAYCODE; TO=PAYROLL.CODE.SOFTWARE
:NEWLINK PAYCODE, PAYROLL.CODE.SOFTWARE
```

NEWGROUP

Creates a new group within an account.

```
NEWGROUP groupname[.acctname][;PASS=[password]]
[;FILES=[filespace]]
[;CPU=[cpu]][;CONNECT=[connect]][;CAP=[capabilitylist]]
[;ACCESS=[fileaccess]]
[;ONVS=volumesetname][;HOMEVS=volumesetname]
NEWGROUP G2.GRIMSBY; CAP=PH,MR
```

NEWJOBQ

Creates a job queue.

```
NEWJOBQ qname[;LIMIT=n]
```

NEWUSER

Defines a new user.

```
NEWUSER username[.acctname][;PASS=[password]]
[;CAP=[capabilitylist]]
[;MAXPRI=[subqueuename]][;LOCATTR=[localattribute]]
[;HOME=[homegroupname]][;UID=[uid]]
NEWUSER LHSMITH;PASS=SMITTY;HOME=HOMEGPX
NEWUSER LHSMITH;UID=120;PASS=SMITTY;HOME=HOMEGPX
```

OCTCOMP

Converts a compiled MPE V/E program into native mode (NM) code for the 900 Series HP 3000.

```
OCTCOMP [input] [,[targetfile][,[list]][;INFO=quotedstring]]
or
OCTCOMP [input] [,[targetfile][,[list][,[INFO=]quotedstring]]]
OCTCOMP SOURCEIN,OCTOUT;INFO="TRANS=1,2,3,4"
```

OPENQ

Opens the spool queue for a specified logical device or device class.

```
OPENQ {ldev   [;SHOW]
       devclass   [;SHOW]
       devname   [;SHOW]
       @   }
OPENQ 6;SHOW
```

OPTION

This command modifies the environment of user defined commands and command files. It is used within the command definition to set up and change the environment dynamically.

```
OPTION [LIST
        NOLIST] [,] [RECURSION
                     NORECURSION]
OPTION LIST
```

OUTFENCE

Defines the minimum priority that an output spoolfile needs in order to be printed.

```
OUTFENCE outputpriority [;LDEV=ldev] [;DEV= {dev
                                             devclass
                                             devname}]

OUTFENCE 14
OUTFENCE 7;LDEV=6
```

PASCAL

Compiles a compatibility mode Pascal/V program. The native mode equivalent of this command is PASXL.

```
PASCAL [textfile][,[uslfile][,listfile]][;INFO=quotedstring]
PASCAL PASCSRC,PASCOBJ,LISTFILE
```

PASCALGO

Compiles, prepares, and executes a compatibility mode Pascal/V program. The native mode equivalent of this command is PASXLGO.

```
PASCALGO [textfile][,listfile][;INFO=quotedstring]
PASCALGO PASCSRC,LISTFILE
```

PASCALPREP

Compiles and prepares a compatibility mode Pascal/V program. The native mode equivalent of this command is PASXLLK.

```
PASCALPREP [textfile][,progfile][,listfile][;INFO=quotedstring]
PASCALPREP PASCSRC,PASCPROG
```

PASXL

Compiles an HP Pascal/iX program.

```
PASXL [textfile][,[objectfile][,[listfile][,libfile]]]
[;INFO=quotedstring]
PASXL MAIN, OBJMAIN
PASXL SUB, OBJSUB
LINK FROM=OBJMAIN,OBJSUB;TO=SOMEPROG
RUN SOMEPROG
```

PASXLGO

Compiles, links, and executes an HP Pascal/iX program.

```
PASXLGO [textfile][,[listfile][,[libfile]]][;INFO=quotedstring]
PASXLGO SOURCE,LISTFILE
```

PASXLLK

Compiles and links an HP Pascal/iX program.

```
PASXLLK [textfile][,[progfile][,[listfile][,libfile]]]
[;INFO=quotedstring]
PASXLLK SOURCE,PROG
```

PAUSE

Allows the user to suspend current activity for a specified number of seconds, or until one or more jobs complete.

```
PAUSE [num_seconds] [;JOB=jobid] [;INTERVAL=interval_secs] [;EXIST | WAIT | NOTEXIST]
```

```
STREAM JLOGEND
#J123
PAUSE JOB=!HPLASTJOB
```

PLISTF (UDC)

The PLISTF UDC executes the LISTFILE command to list descriptions of one or more disk files.

NOTE System-defined UDCs are not automatically available. Your System Manager must use the SETCATALOG command to make these UDCs available for your use. For example:

```
SETCATALOG HPPXUDC.PUB.SYS
```

NOTE If the PLISTF UDC is cataloged, it will override the LISTF command.

Syntax

```
PLISTF [fileset] [,format_opt] [;outfile]
```

PREP

Prepares a compatibility mode program from a user subprogram library (USL) file onto a program file.

```
PREP uslfile,progfile[;ZERODB] [;CAP=capabilitylist] [;PMAP]
[;RL=filename]
[;MAXDATA=segsize][;PATCH=patchsize] [;STACK=stacksize]
[;DL=dlsize]
```

```
[;NOSYM] [;FPMAP
         ;NOFPMAP]
PREP USLX,PROGX
SAVE PROGX
```

PREPRUN

Prepares and executes a compiled compatibility mode program.

```
PREPRUN uslfile[,entrypoint] [;NOPRIV][;RL=filename] [;PMAP]
[;NOCB] [;DEBUG]

[;INFO=quotedstring] [;LMAP[;STDIN[*formaldesig
                                  =fileref
                                  $NULL]]]
[;MAXDATA=segsize]
[;PARM=parameternum] [;STDLIST=[*formaldesig
                                  fileref [,NEW]
                                  $NULL]]
[;STACK=stacksize] [;DL=dlsize] [;PATCH=patchsize]

[;LIB={G
       P
       S}] [;NOSYM][{;FPMAP
                    ;NOFPMAP}] [;CAP=capabilitylist]
PREPRUN XUSL;PMAP;LMAP
```

PRINT

Prints the contents of a file.

```
PRINT[ [FILE=]filename] [;[OUT=]outfile][;[START=]m]
[;[END=] n ][;[PAGE=]p][;{UNN NUM}][;NONOM]
PRINT MYFILE;OUT=XXY
PRINT ./posix/doc/print.doc;start=-10
```

PURGE

Deletes a file from the system.

```
PURGE filereference[;TEMP]
[ ;ONLOCKWORD= SELECT | SKIP]
[ ;ONERROR= CONTINUE | QUIT]
[ ;NOAUTOLOCKWORD | LOCKWORD]
[ ;CONFIRM ;NOCONFIRM | ;CONFIRMALL]
[ ;SHOW | ;NOSHOW]
PURGE PFILE,TEMP
PURGE ./posix/DOC/print.doc
```

PURGEACCT

Removes an account and its groups and users from the system directory or from the specified volume set's directory.

```
PURGEACCT acctname [;ONVS=volumesetname]
PURGEACCT ACCT1
ACCT ACCT1 TO BE PURGED? YES
```

PURGEDIR

Purges (unlinks) one or more directories.

Syntax

```
PURGEDIR [dir=]dir_name [; CONFIRM | NOCONFIRM ]
[; TREE | NOTREE | USENAME ] [; SHOW | NOSHOW]
[; SHOWERRORS | NOSHOWERRORS]
PURGEDIR /MYACCT/MYGRP/dir1
PURGEDIR /MYACCT/MYGRP/dir1;NOTREE
PURGEDIR /MYACCT/MYGRP/dir1/
PURGEDIR /MYACCT/MYGRP/dir1 ;TREE
```

PURGEGROUP

Removes a group (and all files belonging to it) from the system, or from the specified volume set directory.

```
PURGEGROUP groupname[.acctname][;ONVS=volumesetname]
PURGEGROUP GROUP1
GROUP GROUP1 TO BE PURGED? YES
```

PURGEJOBQ

Removes a job queue

```
PURGEJOBQ qname
```

PURGELINK

This command removes a link.

```
PURGELINK [LINK=] linkname
PUREGLINK PAYROLL
PURGELINK /dira/scripts
```

PURGEUSER

Removes a user from an account.

```
PURGEUSER user[.acctname]
PURGEUSER USER1
USER USER1 TO BE PURGED? YES
```

RECALL /=RECALL

Displays all pending console REPLY messages.

```
RECALL
=RECALL
RECALL
THE FOLLOWING REPLIES ARE PENDING:
10:05/#J19/15/LDEV # FOR "L00576" ON TAPE1600 (NUM)?
```

REDO

Allows the user to edit and reexecute any command still retained in the command line history stack.

```
REDO [ [CMD=]cmdid][ [;EDIT=]editstring]
REDO 10
```

REFUSE

Disables jobs/sessions and/or data on a designated device.

```
REFUSE [JOBS,][DATA,]ldev
REFUSE DATA,35
```

RELEASE

Releases a file from file access matrix access control. This command does not affect access control defined by lockwords or access control definitions (ACDs). It cannot be used on directories.

The file matrix access is not enforced until the file is secured with the MPE/iX SECURE command.

```
RELEASE filereference
RELEASE FILE1
```

RELLOG

Removes a user logging identifier from the system.

```
RELLOG logid
RELLOG DATALOG
```

RENAME

Changes identity (file name, lockword, and/or group name) of a disk file.

```
RENAME oldfilereference,newfilereference[,TEMP]
RENAME OLDFILE,NEWFILE/LOCKW.NEWG.NEWACCT,TEMP
RENAME FILE2/LOCKA,FILE2/LOCKB
RENAME MYFILE.GROUP1,MYFILE.GROUP2
```

REPLY/=REPLY

Replies to pending resource requests at the console.

```
REPLY pin,reply
=REPLY pin,reply
10:05/#J19/15/LDEV# FOR "NAS" OF TAPE1600 (NUM)?
REPLY 15,7
```

REPORT

Displays accounting information for the logon account and group. Any user may obtain REPORT information about the user's logon group.

```
REPORT [groupset][,listfile][;ONVS=[volumesetname]]
REPORT SOPRM
```

RESET

Cancels file equations.

```
RESET {formaldesignator
       @}
RESET ALPHA
```

RESETACCT

Resets the running counts of CPU time or connect time accumulated by an account and by all groups within that account to zero.

```
RESETACCT [ [@
            acct] [,[CPU
                     CONNECT] ] ]
RESETACCT @,CPU
```

RESETDUMP

Disarms the debug call that is made during abnormal process termination.

```
RESETDUMP
RESETDUMP
```

RESTORE

Returns files that have been stored on magnetic tape to the system.

```
RESTORE [restorefile][;filesetlist][;option[;...]]
```

where *option* is:

```
[;DEV=device][;SHOW[=showoption[,showoption[,...]]]]
[;FILES=maxfiles]

[;{LOCAL
   GROUP=groupname
   ACC[oun]T=accountname}[;...]]
[;CREATE[= {GROUP
            ACCOUNT
            CREATOR} [,...]]]
[;CREATOR[=username]][;{KEEP
                        NOKEEP}] [;{OLDDATE
                                    NEWDATE}]
[;ONERR[or]= {QUIT
              SKIP}]
[;DIRECTORY] [;LISTDIR] [;FCRANGE=filecode/filecode[,...]]
[;VOLSET=volumesetname] [;VOL=volumename]
[;VOLCLASS=volumeclassname]
FILE T;DEV=TAPE
RESTORE *T;@;KEEP;SHOW
```

RESUME

Resumes execution of a suspended operation.

```
RESUME
RESUME
READ PENDING
Return
```

RESUMEJOB

Resumes a suspended job.

```
RESUMEJOB #Jnnn
RESUMEJOB #J68
```

RESUMELOG

Resumes system logging following suspension caused by an error.

```
RESUMELOG
ST/10:43/LOG FILE NUMBER 104 ERROR #46.
LOGGING SUSPENDED.
RESUMELOG
ST/10:45/LOG FILE NUMBER 104. LOGGING RESUMED.
ST/10:45/LOG FILE NUMBER 104 ON.
```

RESUMESPOOL

Resumes suspended spooler output to a spooled device.

```
RESUMESPOOL ldev{ ;BACK [nnn FILES
                         nnn PAGES]
                  ;FORWARD [nnn FILES
                            nnn PAGES]
                  ;BEGINNING }
RESUMESPOOL 6;BEGINNING
```

RETURN

Causes execution to return from the current user command (UDC or command file) to the calling environment.

```
RETURN
RETURN
```

RPG

Compiles an RPG/V program in compatibility mode.

```
RPG [textfile][,[uslfile][,[listfile][,[masterfile][,newfile]]]]
BUILD OBJECT;CODE=USL
RPG SOURCE,OBJECT,LISTFL
```

RPGGO

Compiles, prepares, and executes an RPG/V program in compatibility mode.

```
RPGGO [textfile][,[listfile][,[masterfile][,newfile]]]
RPGGO SOURCE,LISTFL
```

RPGPREP

Compiles and prepares an RPG/V program in compatibility mode.

```
RPGPREP [textfile][,[progfile][,[listfile][,masterfile][,newfile]]]

RPGPREP,COMFL
SAVE $OLDPASS,NUSL
```

RPGXL

Compiles an RPG/XL program.

```
RPGXL [textfile][,[objectfile][,listfile]] [;INFO=quotedstring]

RPGXL RPGSRC,MYRPGOBJ,LISTFILE
```

RPGXLGO

Compiles, links, and executes an RPG/XL program.

```
RPGXLGO [textfile][,listfile]
```

```
RPGXLGO RPGSRC,LISTFILE
```

RPGXLLK

Compiles and links an RPG/XL program.

```
RPGXLLK [textfile][,[progfile][,listfile]]
```

```
RPGXLLK RPGSRC,RPGPROG
```

RUN

Executes a prepared or linked program.

```
RUN progfile[,["]entrypoint["]][;NOPRIV][;LMAP][;DEBUG]
[;MAXDATA=maxstack] [;PARM=parameternum]
[;STACK=stacksize] [;DL=dlsize] [;NMSTACK=nmstacksize]
[;NMHEAP=nmheapsize]
[;LIB= {G
        P
        S}] [;XL="library[, ...]"] [;NOCB]
[;INFO="quotedstring"]
[;UNSAT=[`"]unsatproc[`"]] [;STDIN=[ *formaldesig
                                    fileref
                                    $NULL ]]
[;STDLIST=[ *formaldesig
            fileref [,NEW]
            $NULL ]] [;PRI= {BS
                             CS
                             DS
                             ES} {#}]
```

```
RUN TESTPROG;DEBUG;STDIN=*INFILE;STDLIST=RESULTS,NEW
```

SAVE

Saves a file in the permanent system file domain.

```
SAVE {$OLDPASS,newfilereference
      tempfilereference          }
```

```
SAVE $OLDPASS,PROGFILE
SAVE TEMPFL
SAVE DATAFILE.GROUPX
```

SECURE

Restores file access matrix access control for a file. The RELEASE command suspends file access matrix access control. Enabling the fiel access matrix does not have an immediate effect on file access if the file is protected by an ACD. ACDs override the file access matrix.

```
SECURE filereference
SECURE FILE1
```

or

```
SECURE ./FILE1
```

SEGMENTER

Starts the MPE segmenter.

```
SEGMENTER [listfile]

FILE LISTFL;DEV=LP
SEGMENTER *LISTFL
```

SET

Defines elements of the command interpreter. It also allows a job using a spooled $STDLIST to mark its standard list device for deletion when the job terminates.

```
SET [STDLIST={DELETE
              SAVE}][;ECHO={ON
                            OFF}][;MSG={ON
                                       OFF}]
[;SPEED={300
         1200
         2400
         4800
         9600
         19200
         19.2K}]

!JOB EXAMPLE, USER.TECHPUB,XGROUP
!CONTINUE
!RUN UPDATE.PUB.SYS;PARM=1;MAXDATA=16000
!IF JCW < FATAL THEN
!SET STDLIST=DELETE
!ENDIF
!EOJ
```

SETCATALOG

Causes the command interpreter to search a catalog of user defined commands (UDCs) and to establish a directory entry for each command, or to clear the previous catalog.

```
SETCATALOG [catfilename[,catfilename,...[,catfilename]]][;SHOW]
[;SYSTEM]
[;ACCOUNT][;USER=username[.acctname]][;RESET][;APPEND]
[;DELETE]

SETCATALOG UDCA,UDCB
SETCATALOG UDCA
SETCATALOG UDCB;APPEND
SETCATALOG
```

SETCLOCK

Alters the system time or system time zone.

```
SETCLOCK {DATE= date spec; TIME= time spec [;GRADUAL
                                            ;NOW]
          CORRECTION= correction spec
          TIMEZONE= time zone spec
          ;CANCEL}

:SETCLOCK DATE=07/04/1993;TIME=15:00
:SETCLOCK CORRECTION= +3600
```

SETDUMP

Arms the system DEBUG facility for a process abort.

```
SETDUMP [DB[,ST[,QS]]][;ASCII][;DEBUG="commands"]

SETDUMP
```

SETJCW

Creates or assigns a value to a job control word (JCW) variable.

```
SETJCW jcwname delimiter value [ {+
                                 -} value]

SETJCW CURR1,100
SETJCW CURR1/WARN
SETJCW NEWJCW=LASTJCW + 56
```

SETMSG

Enables or disables the receipt of user or operator messages at the standard list device.

```
SETMSG {OFF
        ON }

SETMSG OFF
SETMSG ON
```

SETVAR

Assigns values to MPE/iX variables.

```
SETVAR varname{ expression
                ,expression
                ;expression}

SETVAR HPPROMPT "!HPUSER.!HPACCOUNT:"
```

SHOWALLOW

Displays which operator commands have been allowed.

```
SHOWALLOW [user.acct
           user.@
           @.acct
           @.@ ]

SHOWALLOW USER.SYS
```

SHOWCATALOG

Displays information about user defined commands (UDCs).

```
SHOWCATALOG [listfile][;USER=username[.acctname]]

SHOWCATALOG ;USER=@.GRIMSBY
```

SHOWCLOCK

Displays information about the system date and time.

```
SHOWCLOCK

:SHOWCLOCK

SYSTEM TIME: FRI, JUL 24, 1987,  8:47:35 AM
CURRENT TIME CORRECTION:  -3428 seconds
TIME ZONE: 7 HOURS 0 MINUTES WESTERN HEMISPHERE
```

SHOWDEV

Reports the status of input/output devices.

```
SHOWDEV [ldev
         classname]
```

```
SHOWDEV 5
```

SHOWIN

Reports the status of input device files.

```
SHOWIN [#Innn
        STATUS
        SP
        item [;item [;...]]]
```

Where *item* is:

```
[DEV=ldev ]   [JOB= {@J
                    @S
                    @
                    [ #]Jnnn
                    [ #]Snnn}]
[ACTIVE
 OPENED
 READY  ]
```

```
SHOWIN JOB=@S;OPENED
```

SHOWJCW

Displays the current state of one or more job control word (JCW) variables.

```
SHOWJCW [jcwname]
```

```
SHOWJCW JCW1
```

SHOWJOB

Displays status information about jobs/sessions.

```
SHOWJOB [ [#]Snnn
          [#]Jnnn
          STATUS
          SCHED
          item[;item[;...]]][;*listfile]
          [;JOBQ]
```

Where *item* is:

```
[JOB={@J
      @S
      @
      [jsname,]username.acctname}] [;INTRO
                                    ;EXEC
                                    ;SUSP
                                    ;WAIT [,N
                                           ,D] ]
```

```
SHOWJOB STATUS
```

SHOWLOG

Displays the number of the system's current log file and the percentage of disk space used.

```
SHOWLOG
```

```
SHOWLOG
```

SHOWLOGSTATUS

Displays status information about currently opened user logging files assigned to a logging identifier.

```
SHOWLOGSTATUS [logid]
```

```
SHOWLOGSTATUS LEN
```

SHOWME

Reports job/session status.

```
SHOWME
```

```
SHOWME
```

SHOWOUT

Displays the status of output device files.

```
SHOWOUT [#Onnn
         STATUS
         SP
         item[;item[;...]]]
```

Where *item* is:

```
[DEV={ldev
      classname}][JOB= {@J
                        @S
                        @
                       [#]Jnnn
                       [#]Snnn }][ACTIVE
                                  OPENED
                                  LOCKED
                                  READY [,N
                                         ,D]]
```

```
SHOWOUT STATUS
SHOWOUT #0111
```

SHOWPROC

Displays information about one or more processes. (**Native Mode**)

```
SHOWPROC [ [;PIN=] {pinspec(pinspec  [,pinspec]...)}
           [;JOB=] {jobspec(jobspec  [,jobspec]...)}] [ ...]
[ [;FORMAT=] SUMMARY | DETAIL]    [;TREE | ;NOTREE]

[;USER | ;ANYUSER][;SYSTEM]
```

SHOWPROC 1;SYSTEM;TREE	*Show ALL processes if user has SM.*
SHOWPROC 42,#J3;TREE	*Show process information for pin 42 and for job 3 and its descendants.*
SHOWPROC JOB=@J;ANYUSER	*Show all jobs to SM or OP user.*
SHOWPROC (150,#P247,211)	*Show process information for pins 150, 247, 211.*

SHOWQ

Displays process scheduling data and the contents of each subqueue. System supervisor (OP) capability is required to use this command.

```
SHOWQ [;ACTIVE] [;STATUS]
```

SHOWQ

SHOWTIME

Prints current time and date.

```
SHOWTIME
```

SHOWTIME

SHOWVAR

Displays specific variable names and their current values.

```
SHOWVAR [varid][,varid]...[,varid][;JOB=jobid] [;HP | USER |ANY]
```

SHOWVAR *firstvariable, secondvariable*

=SHUTDOWN

Initiates a shutdown of MPE/iX.

```
=SHUTDOWN [ system
            terminal
            dtc
            tape
            disk
            network
            other ]
```

CTRL]] A]]
=SHUTDOWN

CTRL]] A]]
=SHUTDOWN dtc

SHUTQ

Closes the spool queue for the specified logical device or device class.

```
SHUTQ {ldev    [;SHOW]
       devclass [;SHOW]
       devname [;SHOW]
       @}
```

SHUTQ @
SHUTQ 6;SHOW

SPEED

Sets the input and output speed for the user's terminal.

```
SPEED newinspeed, newoutspeed
or
SET SPEED = newspeed
```

```
SPEED 240,240
or
SET SPEED=2400
```

SPL

Compiles a compatibility mode SPL/V program.

```
SPL [textfile][,[uslfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring]

SPL SOURCE,OBJECT,LISTFL
SAVE OBJECT
```

SPLGO

Compiles, prepares, and executes a compatibility mode SPL/V program.

```
SPLGO [textfile][,[listfile][,[masterfile][,newfile]]]
[;INFO=quotedstring]

SPLGO SOURCE,LISTFL
```

SPLPREP

Compiles and prepares a compatibility mode SPL/V program.

```
SPLPREP [textfile][,[progfile][,[listfile][,[masterfile][,newfile]]]]
[;INFO=quotedstring]

SPLPREP SFILE,MYPROG
```

SPOOLER

Controls spooler processes.

```
SPOOLER [DEV=]{ldev
               devclass
               devname}
{;SHOW
 ;OPENQ [;SHOW]
 ;SHUTQ [;SHOW]
 ;START [;OPENQ
         ;SHUTQ] [;SHOW]
 ;STOP  [;FINISH
         ;NOW] [;OPENQ
                ;SHUTQ] [;SHOW]
 ;SUSPEND [ [;FINISH
             ;NOW] [;NOKEEP
                    ;KEEP]
           [;OFFSET=[+
                     -]page]
           [;OPENQ
            ;SHUTQ] [;SHOW] ]
 ;RESUME  [;OFFSET=[+
                   -]page]    [;OPENQ
                               ;SHUTQ] [;SHOW]
 ;RELEASE [;OFFSET=[+
                   -]page][;OPENQ
                           ;SHUTQ] [;SHOW]}

SPOOLER dev;SUSPEND;NOKEEP;OFFSET=1
SPOOLER dev;SUSPEND;KEEP;OFFSET=-3
SPOOLER dev;RESUME;OFFSET=-6
SPOOLER LP;SHOW
```

SPOOLF

Allows a qualified user to alter, print, or delete output spoolfile(s). (**Native Mode**)

```
SPOOLF  {[ [IDNAME=] {spoolid
                      (spoolid[,spoolid]. . .)}
           [;ALTER]     [;SELEQ= {[select-eq]
                                 ^indirect_file}]
           [;DEV={ldev
                  devclass
                  devname}]
           [;PRI=outpri] [;COPIES= numcopies]
           [;SPSAVE]  [;DEFER
                       ;UNDEFER]  [;SHOW] ]
         [ [IDNAME=] {fileset
                      (fileset[,fileset]. . .)}
           [;PRINT] [;DEV= {ldev
                           devclass
                           devname}]
           [;PRI=outpri] [;COPIES= numcopies]
           [;SPSAVE]    [;DEFER
                         ;UNDEFER]  [;SHOW] ]
         [ [IDNAME=] {spoolid
                      (spoolid[,spoolid]. . .)}
           [;DELETE] [;SELEQ= {select-eq
                              ^indirect_file}]
           [;SHOW] ] }
```

Where the select equation, *enclosed in square brackets*, has the following syntax:

```
select_eq ::= [equation]

equation ::= {parm{ >
                  >=
                  <
                  <=
                  <>
                  =       } value
             NOT (equation)
             (equation) {AND
                         OR}  (equation) }
```

```
SPOOLF O@;SELEQ=[DEV=16];ALTER;PRI=8;SHOW
```

STARTSESS

Creates a session on the specified device, if the user has programmatic sessions (PS) capability.

```
STARTSESS ldev;[sessionname,]user[/userpass].acct[/acctpass]
[,group[/grouppass]]
[;TERM={termtype}][;TIME=cpusecs][;PRI= {BS
                                         CS
                                         DS
                                         ES}]
[;INPRI=inputpriority
 ;HIPRI]
[;NOWAIT] [;INFO=ciinfo] [;PARM=ciparm]
```

```
STARTSESS 28;USER.GROUP.ACCOUNT
```

STARTSPOOL

Initiates the spooler process for a device.

```
STARTSPOOL [{ldev[;SHUTQ]
             devclass     }]
```

```
STARTSPOOL 6;SHUTQ
```

STOPSPOOL

Terminates spooling to a specified device or device class.

```
STOPSPOOL [{ldev[;OPENQ]
           devclass     }]

STOPSPOOL 6;OPENQ
```

STORE

Copies disk files onto a magnetic tape. Files copied to tape with the STORE command can be recovered with the RESTORE command.

```
STORE [filesetlist][;[storefile][;option[;option[;...]]]]
```

where *option* is:

```
[;SHOW[=showparmlist]] [;ONERROR=recoverytype] [;FILES=maxfiles]
[;DATE<=accdate
 ;DATE>=moddate][;PURGE] [;PROGRESS [=minutes]]
[;FCRANGE=filecode/filecode[,... ]] [;DIRECTORY] [;TRANSPORT]
[;SPLITVS=split_setname[,split_setname[... ]]]
[;ONVS=volumesetname[,volumesetname[... ]]] [;MAXTAPEBUF]
[;COPYACD][;NOACD] [;RENAME]
```

The *filesetlist* parameter has the following form:

```
filesetitem[,filesetitem[...]]
```

where *filesetitem* may be **!!***indirectfile*, **!!^***indirectfile*, *fileset*.

The *fileset* parameter has the following form:

```
filestostore[-filestoexclude[-filestoexclude[-...]]]
```

```
FILE  DEST;DEV=TAPE
STORE @.GP4X;*BACKUP;SHOW;TRANSPORT
FILE SYSLIST;DEV=LP
; SPLITVS = SPLIT_SETNAME[,SPLIT_SETNAME … ]
STORE @.@.@;*REEL;ONVS=VOLMINE
STORE @.GP4X;*BACKUP;SHOW
FILE T;DEV=TAPE
STORE INDFILE;*T   ** or ^INDFILE;*T **
STORE @.GROUP.ACCOUNT
STORE myset[d-e 1-6]
STORE
STORE @.@.@-@.@.SYS;*TAP;SHOW=SECURITY,DATES,LONG&
STORE @.GROUP.ACCOUNT;PURGE
STORE PROG@.VERSION#.PRODACCT=@.@.ARCHIVE:CREATOR;*T;RENAME
```

STREAM

Spools batch jobs or data from a session or job. The optional time-related parameters of the STREAM command may be used to schedule jobs.

```
STREAM [filename] [,char][;AT = timespec]
[;DAY = {day-of-week
         day-of-month
         days-until-month}]
[;DATE = datespec][;IN = [days[, [hours] [,minutes]]]] [;JOBQ=queuename]

STREAM JOBFILE;IN=1,8
```

STREAMS

Enables or disables the STREAMS device. Allows or disallows users to submit job/data streams.

```
STREAMS {ldev
         OFF }
```

```
STREAMS 10
```

SUSPENDSPOOL

Suspends output to a spooled device.

```
SUSPENDSPOOL ldev[;FINISH]
```

```
SUSPENDSPOOL 6;FINISH
```

SWITCHLOG

Closes the current system log file, then creates and opens a new one.

```
SWITCHLOG
```

```
SWITCHLOG
```

SYSGEN

Starts configuration dialog and/or installation tape creation. This command replaces the SYSDUMP command, which is no longer supported.

```
SYSGEN [basegroup][,newgroup][,inputfile][,outputfile]
```

```
SYSGEN CONFIG,NEWCONF,$STDIN,$STDLIST
```

TELL

Sends a message to another session.

```
TELL {[#]Snnn
      [sessionname,]username.acctname
      @
      @.acctname
      @S                              }[ [;]text]
```

```
TELL @.A PLEASE LOG OFF
```

TELLOP

Sends a message to the system console.

```
TELLOP [text]
```

```
TELLOP PLS MOUNT MYTAPE,VERSION 1
```

TUNE

Changes the filter and/or priority limits of circular subqueues.

```
TUNE [mincycle]  [;CQ=qinfo
                  ;DQ=qinfo
                  ;EQ=qinfo] [ ... ]
```

Where *qinfo* is written in the following form:

```
[base  [,[limit][,[min][,[max][,DECAY
                                ,OSCILLATE]]]]]
```

```
TUNE CQ=152,200,,300;DQ=202,238,1000,1000,OSCILLATE
```

UP

Returns a particular device to its normal function on the system; cancels any DOWN command issued for the device. This command does not apply to disks.

```
UP  ldev
```

```
UP 10
```

VMOUNT

Enables or disables the MPE/iX movable volume facility.

```
VMOUNT {ON [,AUTO]
        OFF      } [;ALL]
```

```
VMOUNT OFF;ALL
```

VSCLOSE

Informs the system to close the specified volume set and take it offline. **(Native Mode)**

```
VSCLOSE volumesetname [ [;PARTVS=] {USER
                                    BACKUP}] [;NOW
                                              ;SPLIT]
```

```
VSCLOSE ACCOUNTING_PAYROLL
VSRELEASESYS ACCOUNTING_PAYROLL
VSCLOSE ACCOUNTING_PAYROLL
```

VSOPEN

Reopens a volume set that has been closed with VSCLOSE. The volume set becomes available for use again. **(Native Mode)**

```
VSOPEN volumesetname[ [;PARTVS=] {USER
                                  BACKUP}]
```

```
VSOPEN ACCOUNTING_PAYROLL
```

VSRELEASE

Releases a volume set that was explicitly reserved by the user with VSRESERVE. The equivalent compatibility mode command is DISMOUNT.

```
VSRELEASE [volumesetname]
```

```
VSRELEASE ACCOUNTING_PAYROLL
```

VSRELEASESYS

Negates a previously issued VSRESERVESYS for the specified volume set. The equivalent compatibility mode command is LDISMOUNT.

VSRELEASESYS *volumesetname*

```
VSRELEASESYS ACCOUNTING_PAYROLL
```

VSRESERVE

Notifies the system to keep a particular volume set on line. The equivalent compatiblity mode command is MOUNT.

VSRESERVE [*volumesetname*][;GEN=*genindex*]

```
VSRESERVE ACCOUNTING_PAYROLL
```

VSRESERVESYS

Instructs the system to reserve a volume set online system-wide. The equivalent compatibility mode command is LMOUNT.

VSRESERVESYS *volumesetname*

```
VSRESERVESYS ACCOUNTING_PAYROLL
```

VSTORE

Verifies that the data on a backup media are valid (for example, there are no media errors) and reports errors incurred by STORE when writing the tape. VSTORE only applies to NMSTORE tapes created in native mode. It does not work on tapes created in compatibility mode. *(Native Mode)*

VSTORE *vstorefile* [;[*filesetlist*][*option* [;...]]]

Where *option* has the following format:

```
[;SHOW=[showoption[,showoption[,...]]]]
[;ONERR[OR]={QUIT
            SKIP
            FULL}][;LOCAL][;DIRECTORY]
```

```
VSTORE *T;@.@.@.; SHOW = OFFLINEV
FILE SYSLIST;DEV=LP
FILE T; DEV=TAPE
VSTORE *T; @.@.@
VSTORE *T;@.@.@; SHOW=OFFLINE
```

VSUSER

Lists all users of a currently reserved, mountable volume set.

VSUSER [*volumesetname*]

```
VSUSER
```

WARN

Sends an urgent message to jobs/sessions.

```
WARN {@
      [#]Jnnn
      [#]Snnn
      [jsname,]user.acct} [;message]
```

```
WARN @;THE SYSTEM WILL SHUTDOWN IN 5 MINUTES. PLS LOG OFF.
WARN #S51;LAST CHANCE TO LOG OFF GRACEFULLY.
```

WELCOME

Defines the welcome message.

```
WELCOME [welcfile]
```

```
 WELCOME
#WELCOME TO THE HP3000 COMPUTER SYSTEM.
#FILES WILL BE STORED EACH DAY BETWEEN 6AM AND 7AM.
#Return]]
```

WHILE

Used to control the execution sequence of a job, UDC, or command file.

```
WHILE expression[DO]
```

```
WHILE SETVAR
   .
   .
   .
ENDWHILE
```

XEQ

Executes any program or command file.

```
XEQ filename [parameterlist] *
```

or

```
XEQ filename [;INFO=quotedstring][;PARM=parmvalue] **
```

```
 * for command files
** for program files
```

```
XEQ fcopy
```

2 Utilities

Utilities Descriptions

Brief descriptions of the utilities available for MPE/iX.

ASOCTBL

Use the ASOCTBL utility to distribute operator commands for specific devices to standard MPE/iX users. This utility creates a table that associates users with device classes in a file called ASOCIATE.PUB.SYS. Users gain access to the corresponding device class with the ASSOCIATE command. The user then has exclusive access to the operator commands that control that device until their association is terminated by logging off or issuing the DISASSOCIATE command. In ASOCTBL, > is the prompt.

```
ASOCTBL
>devclass=username.acctname|
```

or

```
RUN ASOCTBL.PUB.SYS
>devclass=username.acctname|
```

AUTOINST

AUTOINST restores the information contained on the FOS and SUBSYS tapes, sets up the necessary account structure, and creates a customized system load tape (SLT).

```
AUTOINST
```

BULDACCT

BULDACCT runs only on MPE/iX. Use it to take a snapshot of the directory structure on the source system, then recreate it on the destination system. Use BULDACCT to migrate a set of accounts from one volume set to another.

BULDACCT has been enhanced to work with MPE/iX hierarchical directory structures. The hierarchical directory information for accounts, groups, and users is written to BULDJOB1. BULDJOB1 contains the commands used to recreate hierarchical directories and the ACDs associated with each of them.

```
BULDACCT;INFO="processing_options"
```

or

```
BULDACCT
BULDACCT:processing_options
```

BUILDINT

Use the BUILDINT utility to build or change compatibility mode (CM) intrinsic disk files. BUILDINT accepts SPL procedure head declarations (OPTION EXTERNAL is required) and optional commands as input data. If no commands are issued, the procedure head declarations are added to the intrinsic file. Any input data that is not a procedure head terminates input; at this point, the program prints a formatted list of all intrinsics and terminates.

RUN BUILDINT.PUB.SYS or **BUILDINT**

CLKUTIL

CLKUTIL reads and sets the hardware clock. The clock is used for timestamps and time displays. It is usually set to Greenwich Mean Time (GMT). CLKUTIL is a standalone utility, and runs only on the physical console at the ISL prompt.

```
ISL> CLKUTIL
```

DEBUG

DEBUG is used primarily by system programmers, who use it to set breakpoints within programs, and to display and modify data stacks and registers. Access through the DEBUG command is available only to users with privileged mode (PM) capability. Nonprivileged users can get limited access with the ;DEBUG option of the RUN command to debug their applications; the DEBUG utility will not allow them privileged access to the system.

DEBUG or **RUN PROGNAME;DEBUG**

CAUTION Normal MPE safeguards are bypassed in privileged mode. When attempting to modify privileged data on disk, it is possible to destroy file integrity, or the MPE operating system itself. Hewlett-Packard is *not* responsible for changes you make to the operating system or system files. For more information, talk to your Hewlett-Packard service representative.

DIRMIG

DIRMIG (The Directory Migration Tool) utility simplifies the migration of your environment from MPE V/E systems to MPE/iX systems. DIRMIG uses an MPE V/E SYSDUMP tape to transport data including the system directory (account structure), UDCs, user logging IDs, user files and information specifically related to user volumes.

DIRMIG or **RUN DIRMIG.PUB.SYS**

DISCFREE

The DISCFREE utility displays information about the system's free disk space, total volume space capacity, and disk allocation for single volumes or for the whole system. It also determines disk volume fragmentation and transient and permanent disk space limits. DISCFREE displays disk allocation data only for mounted MPE/iX volumes, not scratched volumes; use the DSTAT command to identify currently mounted volumes.

DISCFREE or **DISCFREE"[[format]][,*ldev*]"**

or

```
RUN DISCFREE.PUB.SYS;INFO="[<\esc>format][,ldev]"
```

DISCUTIL

DISCUTIL is a standalone utility that you use to request various disk operations. Use it with the RECOVER command of VOLUTIL to save, and subsequently recover, files from a system that has become logically inoperable. This program can be invoked only at the Initial System Load prompt (ISL>).

```
ISL> DISCUTIL
```

DUMP

The MPE/iX utility DUMP takes a snapshot or dump of system memory. It helps you, or HP support personnel, track down problems in system operation. To use, first request a non destructive boot; this saves the machine's hardware state. Then enter the DUMP command; this lets DUMP take control and dump the processor internal memory, main memory, and all allocated secondary storage marked as dumpworthy.

```
ISL>DUMP
```

EDIT/3000

EDIT/3000 creates and manipulates ASCII files. Use EDIT/3000 commands to insert, delete, replace, modify, search for, and manipulate individual characters, strings of characters, or entire lines of characters. EDIT/3000 can be run in interactive or batch mode.

```
EDITOR
```

FCOPY

Use FCOPY to copy and translate files. You identify the input file and output file. You can request one or more optional functions, such as converting data, copying files from other systems, appending files, extracting subsets of files, or displaying binary files in ASCII format.

The FCOPY utility can be copied from the HFS directories into accounts and groups. Files can be opened from HFS directories into existing files in other HFS directories.

```
FCOPY FROM=filename;TO=filename[;options]
```

In the following example, the file /dir1/doc/print.es is copied to the file myfile in the PUB group of the SYS account.

```
FILE FOO=/dir1/doc/print.es|FCOPY from=*FOO; to=myfile.pub.sys|
```

FSCHECK

The file system check utility (FSCHECK) is a native mode program for detecting and repairing inconsistencies found in the file directories and file label tables of the MPE/iX operating systems. It also provides the additional ability to query and display various attributes of these objects. It is a standalone utility and should be the only program running on the system when it is in use.

```
FSCHECK
```

WARNING	**Do not use this utility without proper service center support. Unauthorized use will void you warranty and may cause data loss.**

GENCAT

Use the GENCAT utility to modify a source catalog, or expand a formatted message catalog (for instance, a message catalog in the user's native language). You don't need any special capabilities to use it.

```
GENCAT          or      RUN GENCAT.PUB.SYS
```

I7DB8CNV

I7DB8CNV converts the character data in an IMAGE data base from any Hewlett-Packard 7-bit national substitution set to ROMAN8. The program is a special version of the program DBLOAD.PUB.SYS and the conversion is done as part of a database load. Generally, DBUNLOAD.PUB.SYS and DBUTIL.PUB.SYS,ERASE are invoked before I7DB8CNV.

```
RUN I7DB8CNV.PUB.SYS
```

IOMAP

IOMAP is a standalone utility that identifies the actual configuration of the system and its paths and devices. Use it to map all existing I/O components in the system and to test I/O system components, using IOMAP's selftest and loopback diagnostics. Run IOMAP from the initial system loader (ISL) prompt; it can only be run on a down system. IOMAP shows processor identification (model, identification, processor board revisions, cache sizes, coprocessors, and main memory) and I/O configuration (paths and components for all cards). By comparing this information with system configuration information, you can see if the hardware that is physically available is the same as the hardware that is configured into the system. Also see SYSMAP.

```
ISL> IOMAP
```

KSAMUTIL

Use KSAMUTIL to manage compatibility mode Keyed Sequential Access Method (CM KSAM) files. You can create a CM KSAM file, rename both the data and key files, save a temporary file as a permanent file, clear all data from a file, purge a file, and verify the

contents and access history of an existing file. The file information may be displayed to the terminal or to a printer. KSAMUTIL runs either in session or in batch mode. You can issue MPE/iX commands within KSAMUTIL, if you put a colon (:) in front of the command name.

```
KSAMUTIL                    or      RUN KSAMUTIL.PUB.SYS
```

LANGINST

Use LANGINST to configure language-specific information onto your HP 3000. You must logon as MANAGER.SYS to run LANGINST. You can do the following tasks with LANGINST:

- Add a language to, or remove a language from, the configuration file.
- Display and modify local formats of a configured language.
- Display the languages supported by Hewlett-Packard.
- Display the language currently configured.
- Modify the system default language.

```
LANGINST
```

LINK EDITOR/XL

Link Editor/XL prepares native mode (NM) compiled object files for execution on 900 Series HP 3000 computers. You can also use Link Editor/XL to create and maintain relocatable and executable libraries. To invoke it and use it interactively, enter LINKEDIT at the MPE/iX prompt. Use the RUN command to invoke Link Editor/XL and specify an information string.

```
LINKEDIT
```

or

```
RUN LINKEDIT.PUB.SYS;INFO=infostring
```

LOGGING (Security Auditing)

You can request that the operating system keep records of particular users, as well as particular events. A new log file is begun automatically every time you reboot. You can also request that a new file be started.

LOGTOOL

The System and Memory Log Analysis Tool (LOGTOOL) can manipulate two types of log files: system log files, and the memory log file. Functions on the various system log files include deleting/clearing the files and displaying their contents. Commands are executed immediately after they are received. LOGTOOL is available in multi-user mode, but some functions require a diagnostic security level.

```
SYSDIAG
DUI> RUN LOGTOOL
```

MAKECAT

Use the MAKECAT utility to access, maintain, and change the following message catalogs:

- CATALOG.PUB.SYS, which contains system error messages.
- CICAT.PUB.SYS, which contains the HELP catalog.
- ser-defined catalogs for various applications.

```
RUN MAKECAT.PUB.SYS
```

N7MF8CNV

N7MF8CNV converts data in MPE text and data files, such as EDIT/XL files, from Hewlett-Packard 7-bit national substitution character set to ROMAN8. The user is prompted for language and file type (text or data). For a text file, each record is converted as one field. For a data file, the user will be prompted on each file for the starting position and length of each field (portion of a record) to be converted.

```
N7MF8CNV        or      RUN N7MF8CNV.PUB.SYS
```

NLIOUTIL

NLIOUTIL is used to dynamically activate the Native Language I/O (NLIO) subsystem for Asian and Middle East/African (MEA) peripheral devices (terminals and printers). NLIO is the basic input and output system integrated into the MPE/iX operating system for Native Language Support (NLS). Once activated by NLIOUTIL, properly configured native devices may use the Native Language I/O facility. Also see NMMGR.

```
NLIOUTIL        or      RUN NLIOUTIL.PUB.SYS;INFO=infostring
```

NLUTIL

NLUTIL is a utility program used to verify a variety of Native Language Support (NLS) languages and corresponding character sets available on the operating system. You can have a complete listing printed on the system printer; you can display a table showing the currently configured languages and their character set types.

```
NLUTIL          or      RUN NLUTIL.PUB.SYS
```

NMMGR

The Node Management Services Configuration Manager is a menu-driven utility you use to configure your HP 3000's data communications subsystems.

```
NMMGR
```

OCA

The Object Code Analyzer is an interactive migration utility used primarily to detect migration incompatibilities in compatibility mode applications. When moving from MPE V/E to MPE/iX, OCA is part of the migration tool set, and the output helps you make your migration plan. Run on MPE/iX systems, OCA identifies incompatibilities that could prevent moving applications from compatibility mode (MPE XL CM) to native mode (MPE XL NM).

OCT

The Object Code Translator translates compatibility mode (CM) object code into functionally equivalent HP Precision Architecture (HP-PA) native instructions. OCT appends translations to the end of a destination file. The resulting file can then be executed on either an MPE V/E-based system or an MPE/iX-based system.

```
OCTOMP
```

PATCH

You may only use PATCH on compatibility mode (CM) programs. Use it to access, display, and/or modify a program file's object code without recompiling the program. You can make simple changes to program instructions or to global stack area variables. PATCH requires the memory location of the target program symbols, the beginning locations of each program unit, and the offsets for each line of code from these locations.

```
PATCH          or          RUN PATCH.PUB.SYS|
```

CAUTION PATCH bypasses normal MPE/iX safeguards and will modify the contents of privileged program files. When attempting to modify privileged data on disk, it is possible to destroy file integrity, or the MPE operating system itself. Hewlett-Packard is *not* responsible for changes you make to the operating system or system files. For more information, talk to your Hewlett-Packard service representative.

PXUTIL

The PXUTIL utility is run by the System Manager to perform operations related to the UID/GID databases. The PXUTIL utility requires exclusive access to the databases. The main function of PXUTIL is to initially create the UID/GID databases, as well as to synchronize existing database files with the current directory. The utility scans through MPE's directory creating UID entries for all existing users and GID entries for all existing accounts. Pressing **BREAK** during the operation of PXUTIL, aborts the process without affecting the existing HPUID.PUB.SYS or HPGID.PUB.SYS files. The PXUTIL utility opens existing files exclusively, and opens two "new" files with the same names.

```
RUN PXUTIL.PUB.SYS
PXUTIL> update
```

SAINT

SAINT is an interactive utility program that analyzes system libraries to produce executable images known as boot images. Its primary function is to produce a boot image for the operating system.

WARNING	**Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.**

SEGMENTER

SEGMENTER manages and prepares compatibility mode (CM) code segments. You can invoke it directly, with the SEGMENTER command. Use it to manage code segments in USL's (user subprogram libraries), RL's (relocatable libraries) and SL's (segmented libraries) and to group RBM's (relocatable binary modules) into code segments. Invoked indirectly (at PREP time), you can use SEGMENTER to define run-time parameters and to group CM program statements into RBM's and code segments with source program statements.

```
SEGMENTER
```

SLPATCH

SLPATCH displays or modifies the contents of a Segmented Library (SL) file. Also see SEGMENTER. Before using this utility you should be familiar with machine-executable instructions and the internal format of segmented library files in the HP 3000 system environment.

```
SLPATCH          or          RUN SLPATCH.PUB.SYS
```

CAUTION	SLPATCH bypasses normal MPE/iX safeguards and will modify the contents of privileged program files. When attempting to modify privileged data on disk, it is possible to destroy file integrity, or the MPE operating system itself. Hewlett-Packard is *not* responsible for changes you make to the operating system or system files. For more information, talk to your Hewlett-Packard service representative.

SOMPATCH

SOMPATCH is used for binary modification of a native mode spectrum object module (SOM) program or library file. Binary modification is referred to normally as patching. This utility also provides online help for command syntax and function.

WARNING	**Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.**

SORT-MERGE/XL

Use SORT to sort files based on single or multiple key items. You can sort data alphabetically, numerically, or in a collating sequence you define; you can request ascending or descending order. Use MERGE to merge data from two or more sorted files into a single, new file. SORT-MERGE/XL operates from within a program, or as a standalone utility in either interactive or batch mode.

```
SORT          or      RUN SORT.PUB.SYS
MERGE         or      RUN MERGE.PUB.SYS
```

SPUTIL

The Native Mode Spooler Utility Program (SPUTIL) allows you to list, manipulate, and transfer spooled device files (spoolfiles) that are created and maintained by MPE/iX. SPUTIL is an MPE/iX replacement for the MPE CM SPOOK5 program.

SPUTIL opens the formal file designator SPUTIN as its $STDIN(X) and the formal file designator SPUTOUT as its $STDLIST. You may redirect these files as desired with a file equation. However the record width of any redirected SPUTOUT should not be less than 80 bytes; otherwise displays and messages may generate an error when SPUTIL directs them to SPUTOUT.

```
SPUTIL
```

STANDARDS

System bootstrap, initial program load (IPL), and initial system load (ISL) standard provides a standard interface through which any Hewlett-Packard Precision Architecture (PA-RISC) computer can boot any operating system. This standard also provides a common user interface for booting PA-RISC systems.

WARNING	**The use of this information without service center support will void your warranty and may cause data loss.**

STORE/RESTORE

Use STORE/RESTORE to store and restore one or more files and directories to and from tape. Options let you store files for backup, transport, or archiving purposes.

```
STORE fileset[;parameters]          RESTORE storfile[;parameters]
```

SWITCH ASSIST TOOL

The Switch Assist Tool is an interactive utility that makes the job of creating an application with modules written both in native and compatibility modes easier to implement. Output is in the form of PASCAL/iX source code.

```
SWAT            or        RUN SWAT.PUB.SYS
```

SYSGEN

Use SYSGEN to modify your system configuration. Changes are written to disk or to tape. They do not become effective until the system is restarted. SYSGEN has a global module and four configurator modules:

1. Input/Output (I/O) Configurator. Configures local devices.
2. Logging (LOG) Configurator. Configures user and system logging processes.
3. Miscellaneous (MISC) Configurator. Configures miscellaneous items.
4. System File (SYSFILE) Configurator. Changes the list of files dumped to an SLT.

```
 SYSGEN
sysgen>command name
```

SYSMAP

SYSMAP displays the hardware configuration of a system by mapping I/O devices, CPU boards and memory boards. It displays device type, product number, logical device number (*ldev*) and device address. SYSMAP is part of the Online Diagnostics Subsystem. To use SYSMAP, first you enter SYSDIAG and get the Diagnostic User Interface prompt. Then you enter RUN SYSMAP and get ENTER MAP>, the SYSMAP prompt. Also see IOMAP.

```
SYSDIAG
DUI> RUN SYSMAP
ENTER MAP>
```

TERMDSM

Use the TERMDSM tool to diagnose, dump, and reset logical devices, ports, and data communications and terminal controllers (DTCs). TERMDSM also performs status checks of ports and DTC's.

```
SYSDIAG
DUI > RUN TERMDSM
```

tic

The tic utility compiles source terminfo descriptions. The compiled entry is installed under the /usr/lib/terminfo directory hierarchy. If the TERMINFO environment variable is set, results are placed in the directory it points to instead. Entries are stored in directories that match the first character of their name. The entry for the VT-100 terminal, for example, is stored in /usr/lib/terminfo/v/vt100.

```
tic.hpbin.sys /product/curses/lib/terminfo/ansi
```

TTUTIL

TTUTIL is a screen-driven program that lets you modify characteristics of serial port connections (such as flow control, modem control, printer control and character handling) by modifying the terminal type file assigned to the port. You can create, modify or view an existing terminal or printer type file..

RUN TTUTIL.PUB.SYS or **TTUTIL**

untic

The untic utility decompiles a terminfo binary file into its source format. If a TERMINFO environment variable is set, the untic utility searches the specified directory; otherwise, untic assumes the file is in the directory /usr/lib/terminfo. The output of an untic decompile is sent to the standard output

untic.hpbin.sys ansi

V7FF8CNV

In VPLUS/XL forms files, V7FF8CNV converts text and literals from a Hewlett-Packard 7-bit national substitution character set to ROMAN8 character set.

V7FF8CNV or **RUN V7FF8CNV.PUB.SYS**

VERSION

VERSION is a native mode utility program that displays program file information. For compatibility mode (CM) program files, it displays segment, stack, data reference base, and capabilities. For native mode (NM) executable files, it displays information on procedures, libraries, capabilities, stack, heap, entry names, and $version strings. ($version string information is displayed for NM object files and nonexecutable library files.) If VERSION is invoked without a file name or a file set for input, the VERSION> prompt continues until EXIT or a colon (:) is entered. If the input to VERSION is a file set, every file in the set will be processed even if an error occurs processing a previous file. If there is an error opening a file, the file system error will be displayed in addition to the VERSION error message.

VERSION or **VERSION** *filename*
or **VERSION "***filename* **[,***search string***]"**

The *search string* is the name of a particular $version string in a system object module SOM. (Not applicable for CM program files.) The quotes are required if a search string is specified.

VOLUTIL

Use VOLUTIL commands to manipulate volume sets: to manage and maintain individual volumes, volume sets, and volume classes, and to make inquiries about their contents, availability, and status. You can use any MPE/iX system command from within VOLUTIL by entering a colon (:) before the command name. VOLUTIL commands are organized into four groups:

- Commands that operate on sets and end with 'SET'.
- Commands that deal with classes and end with 'CLASS'.
- Commands that control volumes and end with 'VOL'.
- Miscellaneous commands.

```
VOLUTIL
volutil> command name
```

or

```
RUN VOLUTIL.PUB.SYS
volutil> command name
```

3 Intrinsics Descriptions

Descriptions of the Intrinsics Available in MPE/iX

Alphabetical listing of all Intrinsics available.

ABORTSESS

NM and CM callable.

Enables a program to abort a specified job or session from the system.

```
            I16V I32V   I16A
ABORTSESS(jsid,jsnum,jsstatus);
```

ACTIVATE

NM and CM callable.

Activates a newly created process, or a process suspended with the SUSPEND intrinsic. Requires process handling (PH) capability.

```
          I16V U16V
ACTIVATE(pin,allow);
```

ADJUSTUSLF

NM and CM callable.

Adjusts directory space in a USL file by moving the start of the information block forward (or backward) on a user subprogram library (USL) file, thereby increasing (or decreasing) the space available for the file directory block. The overall length of the file does not change. This intrinsic is intended for programmers writing compilers. A USL contains CM object code and is meaningful only in the CM program development process.

```
   I16                 I16V     I16V
uslferror:=ADJUSTUSLF(uslfnum,adjustment);
```

ALMANAC

NM and CM callable.

Returns the numeric date information for a date returned by the CALENDAR intrinsic. The returned information is year of century, month of year, day of month, and day of week.

```
        U16V  U16A      I16      I16       I16      I16
ALMANAC(date,daterror,yearnum,monthnum,daynum,weekdaynum);
```

ALTDSEG

NM and CM callable.

Reduces the storage required by the extra data segment when moved into main memory and expands storage as required, allowing for a more efficient use of memory. Data segment management (DS) capability is required. Data segment management (DS) intrinsics are not recommended for use in the MPE/iX native mode programming environment; use of DS intrinsics degrade program performance.

```
          U16V   I16V   I16
ALTDSEG(index,increment,size);
```

ARITRAP

NM and CM callable.

Collectively enables all arithmetic traps (except the IEEE inexact result trap) or disables all arithmetic traps.

```
          I*V
ARITRAP(trapstate);
```

ASCII

NM and CM callable.

Converts a 16-bit binary number to a specified base and represents it as a numeric ASCII string.

```
  I16             *     I16V   CA
numchar:=ASCII(binvalue,base,asciieqv);
```

BEGINLOG

NM and CM callable.

Posts a special record to the user logging file to mark the beginning of a logical transaction. When BEGINLOG is called, the logging memory buffer is flushed to ensure that the record gets to the logging file. User logging (LG) or system supervisor (OP) capability is required.

```
          I32  U16A  I16   I16    I16
BEGINLOG(index,data,length,mode,logstatus)
```

BINARY

NM and CM callable.

Converts a numeric (octal or decimal) ASCII string to a 16-bit twos complement binary value.

```
 I16            CA     I16V
bineqv:=BINARY(asciieqv,length);
```

CALENDAR

NM and CM callable.

Returns the calendar date, including the day of year and the year of century.

```
U16
date:=CALENDAR;
```

CATCLOSE

NM and CM callable.

Closes an application message catalog that was opened with CATOPEN.

```
            I32      U16A
CATCLOSE(catindex,catstatus)
```

CATOPEN

NM and CM callable.

Opens an application message catalog that was formatted with the GENCAT utility. CATOPEN returns a value that identifies the catalog and is used by CATREAD and CATCLOSE.

```
  I32                  CA       U16A
catindex:=CATOPEN(formaldesig,catstatus);
```

CATREAD

NM and CM callable.

Provides access to messages in an application message facility formatted by the GENCAT utility. The CATOPEN intrinsic opens the message catalog.

```
   I16                I32V     I16V    I16V     U16A
msglength:=CATREAD(catindex,setnum,msgnum,, catstatus,
                       CA     I16V     CA     CA     CA    CA     CA    I16V
                   buffer,buffersize,parm1,parm2,parm3,parm4,parm5,msgdest);
```

CAUSEBREAK

NM and CM callable.

Interrupts the program (the entire process structure). The CAUSEBREAK intrinsic is the programmatic equivalent to pressing **Break** in a session. It is not applicable in jobs. The program is suspended while in break mode. Execution of the program resumes where the interruption occurred if you enter the RESUME command, or aborts if you enter the ABORT command.

```
CAUSEBREAK;
```

CLEANUSL

NM and CM callable.

Deletes all inactive entries from currently managed USL files and returns the file number of the new USL file. Therefore, you must test the condition code immediately upon return from the intrinsic. Unpredictable results occur if an error number is used as a file number. A USL contains CM object code and is meaningful in the CM program development process only.

```
      I16               I16V        CA
filenum:=CLEANUSL(uslfnum,formaldesig);
```

CLOCK

NM and CM callable.

Returns the time (hours, minutes, seconds, and tenths of seconds) according to the system timer.

```
I32
time:=CLOCK;
```

CLOSELOG

NM and CM callable.

Closes access to the user logging facility. User logging (LG) or system supervisor (OP) capabilities are required.

```
          I32   I16    I16
CLOSELOG(index,mode,logstatus);
```

COMMAND

NM and CM callable.

Executes an MPE/iX command programmatically.

```
           CA       I16     I16
COMMAND(cmdimage,cmderror,parmnum);
```

CREATE

NM and CM callable.

Creates a process as a child of the calling process. Process handling (PH) capability is required.

```
            CA          CA   I16 I16V U16V
CREATE(formaldesig,entryname,pin,parm,loadflag,
         I16V  I16V   I16V    U16V    I16V
       stacksize,dlsize,maxdata,priorityclass,rank);
```

CREATEPROCESS

NM and CM callable.

Creates a process and allows you to assign $STDIN and $STDLIST to any file. Process handling (PH) capability is required.

```
                  I*       I16     CA      I32A   I32A
CREATEPROCESS(createstatus,pin,formaldesig,itemnum,item);
```

createstatus is a 32-bit signed integer by reference in Native Mode (NM), and a 16-bit signed integer by reference for Compatibility Mode (CM).

CTRANSLATE

NM and CM callable.

Converts a string of characters between EBCDIC and ASCII, or between EBCDIK (HP-specific version of EBCDIC) and KANA8 (8-bit, Japanese International Standard (JIS) version of USASCII code).

```
              I16V      CA      CA        I16V       CA
CTRANSLATE(transcode,inbuffer,outbuffer,bufferlength,transtable);
```

DASCII

NM and CM callable.

Converts a 32-bit binary number to a specified base and represents it as a numeric ASCII string.

```
  I16             I32V   I16V   CA
numchar:=DASCII(binvalue,base,asciieqv);
```

DATELINE

NM and CM callable.

Returns the current date and time, including the day of week, month, day, year, hours, and minutes.

```
             CA
DATELINE(datebuffer);
```

DBINARY

NM and CM callable.

Converts a numeric ASCII string to a 32-bit binary value. The numeric ASCII string can be octal, hexadecimal, or decimal.

```
  I32             CA     I16V
dbineqv:=DBINARY(dasciieqv,length);
```

DEBUG

NM and CM callable.

Invokes the debug facility from an interactive program and allows object code to be analyzed. Consult the *MPE/iX System Debug Reference Manual* (32650-90013) before attempting to use the debug facility.

```
DEBUG;
```

DLSIZE

NM and CM callable.

Causes the area between DL and DB in the compatibility mode (CM) stack to be expanded or contracted within the CM stack segment.

```
  I16          I16V
dldbsize:=DLSIZE(size);
```

DMOVIN

NM and CM callable.

Copies data from an extra data segment into a data area. Data segment management (DS) capability is required. Data segment management (DS) intrinsics are not recommended for use in the NM programming environment; use of DS intrinsics in NM degrades an NM program's performance.

```
        U16V     I16V      I16V    U16A
DMOVIN(index,displacement,number,location);
```

DMOVOUT

NM and CM callable.

Copies data from the data area to an extra data segment. Data segment management (DS) capability is required. Data segment management (DS) intrinsics are not recommended for use in the NM programming environment; use of DS intrinsics in NM degrades the NM program's performance.

```
         U16V     I16V      I16V    U16A
DMOVOUT(index,displacement,number,location);
```

ENDLOG

NM and CM callable.

Posts a record to the logging file marking the end of a logical transaction. When the record is posted, ENDLOG flushes the user logging memory buffer to ensure that the record gets to the logging file. User logging (LG) or system supervisor (OP) capability is required.

```
        I32  U16A  I16   I16    I16
ENDLOG(index,data,length,mode,logstatus);
```

EXPANDUSLF

NM and CM callable.

Changes length of a USL file by creating a USL file with the **increment** length longer or shorter than the USL file specified by **uslfnum**. The old USL file is copied to the new file with the same file name; the old USL file is then deleted. A USL contains CM object code and is meaningful only in the CM program development process.

```
  I16                 I16V     I16V
filenum:=EXPANDUSLF(uslfnum,increment);
```

FATHER

NM and CM callable.

Returns the process identification number (PIN) of the parent calling process. Process handling (PH) capability is required.

```
I16
pin:=FATHER;
```

FCHECK

NM and CM callable.

Returns specific details about error conditions that occurred when a file system intrinsic returns a condition code indicating an I/O error. FCHECK applies to files on any device.

```
        I16V    I16     I16     I32     I16
FCHECK(filenum,fserrorcode,translog,blocknum,numrecs);
```

FCLOSE

NM and CM callable.

Terminates access to a file on any device by closing the reference file descriptor. If the file is not being accessed by another process, resources associated with the open file description are released.

```
          I16V     I16V       I16V
FCLOSE(filenum,disposition,securitycode);
```

FCONTROL

NM and CM callable.

Performs various control operations on a file or on the device where the file resides, including:

- Supplying a printer or terminal carriage control directive.
- Verifying I/O.
- Reading the hardware status word for the device where the file resides.
- Setting a terminal's timeout interval.
- Repositioning a file at its beginning.
- Writing an end-of-file marker.
- Skipping forward or backward to a tape mark.

```
            I16V    I16V  *
FCONTROL(filenum,itemnum,item);
```

FDELETE

NM and CM callable.

Deactivates a specified logical record in an RIO file.

```
         I16V    I32V
FDELETE(filenum,lrecnum);
```

FDEVICECONTROL

NM and CM callable.

Provides control operations to a printer, terminal, or a spooled device file and is used to:

- Download character sets, forms, and internal or control tables used in printing.
- Control the page size, pen positioning, form and use of character sets, the number of copies to be printed, and all other printing environment characteristics.
- Perform control operations on a terminal, printer, or spooled device file.

```
                I16V    UDS   I16V      I16V    U16V   U16V     U16
FDEVICECONTROL(filenum,buffer,length,controlcode,parm1,parm2,fserrorcode);
```

FERRMSG

NM and CM callable.

Returns a message corresponding to an FCHECK error number and enables error messages to be displayed from a program.

```
              I16        CA        I16
FERRMSG(fserrorcode,msgbuffer,msglength);
```

FFILEINFO

NM and CM callable.

Returns information about a file.

```
            I16V      I16V     *
FFILEINFO(filenum[,itemnum,item] [...]);
```

Up to five *itemnum/item* pairs can be specified.

FFINDBYKEY

NM and CM callable.

Positions the record pointer at the beginning of the first record matching the key value comparison. For KSAM files only.

```
                I16V   CA    I16V   I16V  I16V
FFINDBYKEY(filenum,value,location,length,relop);
```

FFINDN

NM and CM callable.

Positions the logical record pointer to the relative record number according to the key sequence. For KSAM files only.

```
              I16V  DV    I16V
FFINDN(filenum,number,location);
```

FGETINFO

NM and CM callable.

Returns access and status information about a file. FGETINFO is provided for compatibility with MPE V/E-based systems only. It is recommended that FFILEINFO be used to access data.

```
           I16V       CA      U16   U16    I16    I16
FGETINFO(filenum,formaldesig,foption,aoption,lrecsize,devtype,

 U16     U16   I16   I32  I32 I32    I32    I32
ldevnum,hdaddr,filecode,lrecptr,eof,filelimit,logcount,physcount,

 I16   U16   I16      I16        CA     I32
blksize,extsize,numextent,userlabels,creatorid,labaddr);
```

FGETKEYINFO

NM and CM callable.

Requests access and status information about a KSAM file. For KSAM files only.

```
                I16V     BA     BA
FGETKEYINFO(filenum,param,control)
```

FINDJCW

NM and CM callable.

Searches the job control word table for a specified job control word (JCW) and returns its value.

```
             CA       U16       I16
FINDJCW(jcwname,jcwvalue,jcwstatus);
```

FINTEXIT

NM and CM callable.

Causes the return from your interrupt procedure.

```
               U16V
FINTEXIT(interruptstate);
```

FINTSTATE

NM and CM callable.

Enables/disables all software interrupts against the calling process.

```
  U16                    U16V
oldstate:=FINTSTATE(interruptstate);
```

FLABELINFO

NM and CM callable.

Returns information from the file label of a disk file.

```
                CA       I16V      I16        I16A    REC    I16A
FLABELINFO(formaldesig,mode,fserrorcode,itemnum,item,itemerror);
```

FLOCK

NM and CM callable.

Dynamically locks a file. If dynamically locking more than one RIN, multiple RIN (MR) capability is required.

```
          I16V   U16V
FLOCK(filenum,lockflag);
```

FLUSHLOG

NM and CM callable.

Flushes the contents of the user logging memory buffer to the user logging file. User logging (LG) or system supervisor (OP) capability is required.

```
          I32  I16
FLUSHLOG(index,logstatus);
```

FMTCALENDAR

NM and CM callable.

Passes any calendar date, in the same format as the CALENDAR intrinsic, and returns it in the following format: FRI, JAN 27, 1989

```
            U16V    CA
FMTCALENDAR(date,formatdate);
```

FMTCLOCK

NM and CM callable.

Passes the time of day, in the same format as the CLOCK intrinsic, and returns it in the following format:

```
12:39 AM

         I32V    CA
FMTCLOCK(time,formattime);
```

FMTDATE

NM and CM callable.

Passes in the calendar date and time of day, in the same format as the CALENDAR and CLOCK intrinsics, and returns it in the following format:

```
FRI, JAN 27, 1989, 12:39 AM

         U16V I32V   CA
FMTDATE(date,time,datetime);
```

FOPEN

NM and CM callable.

Establishes access to a file and defines the physical characteristics of the file prior to access.

```
   I16              CA    U16V   U16V  I16V   CA    CA    I16V
filenum:=FOPEN(formaldesig,foption,aoption,recsize,device,formmsg,userlabels,
             I16V    I16V    I32V   I16V    I16V   I16V
        blockfactor,numbuffer,filesize,numextent,initialloc,filecode);
```

FPARSE

NM and CM callable.

Parses and validates MPE (only) file designators.

```
            CA       I16A U16A I32A
FPARSE(formaldesig,result,item,vector);
```

FPOINT

NM and CM callable.

Sets the logical record pointer for a disk file containing fixed-length or undefined-length records to any logical record. When the next FREAD or FWRITE file request is made, this record is read or written to.

(KSAM) Sets both the chronological and logical record pointers to the next record in chronological sequence (the order records were written to the file).

```
          I16V    I32V
FPOINT(filenum,lrecnum);
```

FREAD

NM and CM callable.

Reads a logical record or portion of a record from a file to the stack.

```
      I16             I16V   UDS    I16V
transfercount:=FREAD(filenum,buffer,length);
```

FREADBACKWARD

NM and CM callable.

Reads a logical record backward from the current record pointer. Data is presented as if read forward. Used for tape files only. Can recover tape errors when handling I/O management and data recovery routines.

```
      I16                   I16V   UDS   I16V
transfercount:=FREADBACKWARD(filenum,buffer,length);
```

FREADBYKEY

NM and CM callable.

Reads a logical record randomly from a KSAM file to the data stack. For KSAM file only.

```
  I16V               I16V    LA   I16V   CA    I16V
length:=FREADBYKEY(filenum,target,tcount,value,location);
```

FREADC

NM and CM callable.

Reads a logical record in chronological sequence from a KSAM file to the data stack. For KSAM files only.

```
  I16V           I16V    LA   I16V
length:=FREADC(filenum,target,tcount);
```

FREADDIR

NM and CM callable.

Reads a specific logical record or portion of a record from a direct-access disk file to the data stack.

```
              I16V    UDS    I16V     I32V
FREADDIR(filenum,buffer,length,lrecnum);
```

FREADLABEL

NM and CM callable.

Reads a user-defined label from a disk or magnetic tape file.

```
                 I16V    UDS    I16V  I16V
FREADLABEL(filenum,buffer,length,labelid);
```

FREADSEEK

NM and CM callable.

Moves a record from a disk file to a buffer in anticipation of a FREADDIR intrinsic call.

```
               I16V    I32V
FREADSEEK(filenum,lrecnum);
```

FREEDSEG

NM and CM callable.

Releases an extra data segment assigned it by the GETDSEG intrinsic. Data segment management (DS) capability is required. Data segment management (DS) intrinsics are not recommended for use in the MPE/iX native mode programming environment. Use of DS intrinsics in NM will degrade your program's performance.

```
           U16V U16V
FREEDSEG(index,id);
```

FREELOCRIN

NM and CM callable.

Frees all local resource identification numbers (RINs) from allocation to a job/session.

```
FREELOCRIN;
```

FRELATE

NM and CM callable.

Determines whether a file pair (on any device) is interactive, duplicative, or both interactive and duplicative.

```
  U16                    I16V       I16V
intordup:=FRELATE(infilenum,listfilenum);
```

FREMOVE

NM and CM callable.

Marks the current record in a KSAM file for deletion. For KSAM files only.

```
              I16V
FREMOVE(filenum)
```

FRENAME

NM and CM callable.

Renames an open disk file (and its lockword, if applicable). The file being renamed must be either:

- A new file.
- An old file (permanent or temporary), opened for exclusive access with the *exclusive* option of the HPFOPEN/FOPEN intrinsics, and with security provisions allowing write access.

```
          I16V       CA
FRENAME(filenum,formaldesig);
```

FSETMODE

NM and CM callable.

Controls the following access modes of files or devices:

- Issuing carriage return and line feed to terminal after a terminal read.
- Reporting tape automatic error recovery.
- Guaranteeing chronological order of user program write requests.
- Blocking program execution until physical completion of write requests.

```
            I16V      U16V
FSETMODE(filenum,modeflags);
```

FSPACE

NM and CM callable.

Moves a record pointer forward or backward on a magnetic tape or disk file, spaces physical records on magnetic tape files and logical records on disk files.

```
          I16V       I16V
FSPACE(filenum,displacement);
```

FUNLOCK

NM and CM callable.

Dynamically unlocks a file's global resource identification number (RIN) that was locked with the FLOCK intrinsic.

```
          I16V
FUNLOCK(filenum);
```

FUPDATE

NM and CM callable.

Updates (writes) a logical record in a disk file.

```
              I16V   UDS   I16V
FUPDATE(filenum,buffer,length);
```

FWRITE

NM and CM callable.

Writes a logical or physical record or portion of a record from the stack to a file on any device.

```
             I16V   UDS    I16V     U16V
FWRITE(filenum,buffer,length,controlcode);
```

FWRITEDIR

NM and CM callable.

Writes a specific logical record from the stack to a disk file.

```
                I16V   UDS    I16V   I32V
FWRITEDIR(filenum,buffer,length,lrecnum);
```

FWRITELABEL

NM and CM callable.

Writes a user-defined label onto a disk file or magnetic tape file that is labeled with an ANSI-standard or IBM-standard label. It also overwrites old user labels.

```
                  I16V   UDS   I16V  I16V
FWRITELABEL(filenum,buffer,length,labelid);
```

GENMESSAGE

NM and CM callable.

Provides access to messages in catalogs that were formatted with the MAKECAT utility.

```
      I16              I16V   I16V    I16V   CA   I16V    I16V
msglength:=GENMESSAGE(filenum,setnum,msgnum,buffer,buffersize,parmask,
               *       *      *       *       *       I16V       I16
            param1,param2,param3,param4,param5,msgdestination,errornum);
```

GETDSEG

NM and CM callable.

Creates or acquires an extra data segment for use by the process. Data segment management (DS) capability is required. Data segment management (DS) intrinsics are not recommended for use in the MPE/iX native mode programming environment. Use of DS intrinsics in NM degrades your program's performance.

```
            U16   I16  U16V
GETDSEG(index,length,id);
```

GETINFO

NM and CM callable.

Returns user-supplied information that was passed to a process when it was created.

```
 I16              CA      I16     I16
result:=GETINFO(infostring,infolength,parm);
```

GETJCW

NM and CM callable.

Returns the value of the system-defined job control word (JCW) to the calling process.

```
U16
jcw:=GETJCW;
```

GETLOCRIN

NM and CM callable.

Acquires local resource identification numbers (RINs) for a job/session.

```
             U16V
GETLOCRIN(rincount);
```

GETORIGIN

NM and CM callable.

Returns the source of the activation call for the calling process that has been previously suspended and subsequently reactivated. The source of the activation request can be the parent process, a child process, or another source (for example, an interrupt or the timer). Process handling (PH) capability is required.

```
 I16
source:=GETORIGIN;
```

GETPRIORITY

NM and CM callable.

Changes the priority of a process. Process handling (PH) capability is required.

```
             I16V   U16V       I16V
GETPRIORITY(pin,priorityclass,rank);
```

GETRIVMODE

NM and CM callable.

Dynamically enters privileged mode. Privileged mode (PM) capability is required. The normal checks and limitations that apply to the standard users in MPE/iX are bypassed in privileged mode (PM). It is possible for a PM program to destroy file integrity, including the MPE/iX operating system software itself. Hewlett-Packard will investigate and attempt to resolve problems resulting from the use of PM code. This service, which is not provided under the standard service contract, is available on a time and materials billing basis. Hewlett-Packard will not support, correct, or attend to any modification of the MPE operating system software.

```
GETPRIVMODE;
```

GETPROCID

NM and CM callable.

Returns the process identification number (PIN) of a child process. Process handling (PH) capability is required.

```
I16              I16V
pin:=GETPROCID(numchild);
```

GETPROCINFO

NM and CM callable.

Returns status information about the parent or a child process. Process handling (PH) capability is required.

```
   I32                 I16V
processinfo:=GETPROCINFO(pin);
```

GETUSERMODE

NM and CM callable.

Dynamically returns a program to nonprivileged mode.

```
GETUSERMODE;
```

HP32208

CM callable only.

Returns the current VUF (version, update, fix level) of KSAM/3000.

```
   D
version:=HP32208
```

HPACDINFO

Lists security information from the access control definition (ACD) of a specified file or device. Any user with RACD access to an ACD can obtain information about that ACD.

Syntax

I32 IV * HPACDINFO(**status**,**itemnum1**,**item1** IV * [,*itemnum2*,*item2*][,...]);

HPACDPUT

Manipulates security information in the access control definition (ACD) of a specified file or device.

Syntax

I32 IV * IV * HPACDPUT(**status,itemnum1,item1,itemnum2,item2**);

HPCICOMMAND

NM callable only.

Executes a command programmatically.

```
                CA       I16      I16    I16V
HPCICOMMAND(cmdimage,cmderror,parmnum,msglevel);
```

HPCIDELETEVAR

NM callable only.

Removes a valid variable name from the session-level variable table.

```
                   CA     I32
HPCIDELETEVAR(varname,status);
```

HPCIGETVAR

NM callable only.

Retrieves a valid variable name from the session-level variable table and returns the current value and/or attributes.

```
               CA     I32    U32     *
HPCIGETVAR(varname,status[,itemnum,item] [...])
```

Up to six *itemnum*/*item* pairs can be specified.

HPCIPUTVAR

NM callable only.

Sets the value of a session-level variable.

```
               CA     I32     U32    *
HPCIPUTVAR(varname,status[,itemnum,item] [...])
```

Up to three *itemnum/item* pairs can be specified.

HPDEBUG

NM callable only.

Enters the system debugger and optionally executes a defined set of system debug commands.

```
          I32    CA     I32V   *
HPDEBUG(status,cmdstr[,itemnum,item][...]);
```

HPDEVCONTROL

NM callable only.

Provides access to specified peripheral functionality without the device being opened. Allows access to device utilities; not for general control (for example, reading or writing). Nonshareable device (ND) capability is required.

```
                I32   CA     I32   I32
HPDEVCONTROL(status,ldev,itemnum,item);
```

HPENBLTRAP

NM callable only.

Selectively enables or disables arithmetic traps.

```
            I32V   I32
HPENBLTRAP(mask,oldmask);
```

HPERRDEPTH

NM callable only.

Returns the current depth of the process error stack.

```
             I32   I32
HPERRDEPTH(depth,status);
```

HPERRMSG

NM callable only.

Obtains or displays error messages from the system catalog.

```
             I32V     I32V   I16      I32V    CA    I16   I32
HPERRMSG(displaycode,depth,errorproc,errornum,buffer,buflength,status);
```

HPERRREAD

NM callable only.

Reads any specified error from the process stack.

```
            I32V   I32       I32   I32
HPERRREAD(depth,errornum,procnum,status
```

HPFDUPLICATE

NM callable only.

Creates duplicate file descriptors for files opened for MULTI, SHARED, or EXCLUSIVE access.

Syntax

```
  I16           I32    I32V
*filenum:=HPFDUPLICATE(source,status,target);
```

HPFIRSTLIBRARY

NM callable only.

Returns the file name of the first native mode executable library (XL) in the binding sequence of the calling process.

```
                  CA      I32   I32
HPFIRSTLIBRARY(formaldesig,status,length);
```

HPFOPEN

NM callable only.

Establishes access to a file on any device and creates a file on any shareable device.

```
           I32    I32    I32V    *
HPFOPEN(filenum,status[,itemnum,item] [...]);
```

Up to 41 *itemnum/item* pairs can be specified.

HPFPCONVERT

NM callable only.

Converts data between binary floating-point formats.

```
              *        *        I16V    I16V   I32    I16     I16V
HPFPCONVERT(source,destination,sformat,dformat,status,exceptions,roundmode)
```

HPGETPROCPLABEL

NM callable only.

Dynamically loads a native mode (NM) executable library procedure.

```
                    CA      U32    I32    CA       B
HPGETPROCPLABEL (procname,plabel,status,firstfile,casesensitive);
```

HPLOADCMPROCEDURE

NM callable only.

Obtains CM procedure plabel in preparation for Switch to CM through plabel.

```
 U16                          CA    U16V    I32
plabel:=HPLOADCMPROCEDURE(procname,library,status);
```

HPLOADNMPROC

CM callable only.

Returns the plabel of an NM procedure.

```
 U32                  CA      I16V     CA     I16V
plabel:=HPLOADNMPROC(procname,proclen,libname,liblen);
```

HPMERGEEND

NM callable only.

Releases the MERGE/XL work area and ends the merging operation.

```
             I32    I32A
HPMERGEEND(status,statistics);
```

HPMERGEERRORMESS

NM callable only.

Accepts HP MERGE intrinsic error code values and returns the error messages associated with them.

```
                    I32     CA     I32
HPMERGEERRORMESS(status,message,length);
```

HPMERGEINIT

NM callable only.

Initializes the MERGE/XL subsystem.

```
              I32    I32A      PROC      I32A      PROC
HPMERGEINIT(status,inputfiles,preprocessor,outputfiles,postprocessor,

  32V     I32V  I32A  CA     PROC      PROC    I32A     I32V    I32A
keysonly,numkeys,keys,altseq,keycompare,errorproc,statistics,memsize,charseq);
```

HPMERGEOUTPUT

NM callable only.

Retrieves records, one at a time, from MERGE/XL.

```
                 I32   CA   I32
HPMERGEOUTPUT(status,buffer,length);
```

HPMERGESTAT

NM callable only.

Prints MERGE/XL statistics on $STDLIST.

```
               I32   I32A
HPMERGESTAT(status,statistics);
```

HPMERGETITLE

NM callable only.

Prints the version number and title information for MERGE/XL on $STDLIST.

```
                I32
HPMERGETITLE(status);
```

HPMYFILE

NM callable only.

Returns the file name of the native mode program or executable library (XL) that called the HPMYFILE intrinsic.

```
              CA       I32  I32
HPMYFILE(formaldesig,status,length);
```

HPMYPROGRAM

NM callable only.

Returns the file name of the program being executed by the calling process.

```
                    CA        I32  I32
HPMYPROGRAM(formaldesig,status,length);
```

HPRESETDUMP

NM callable only.

Disarms the system debugger call from a process abort.

```
                I32
HPRESETDUMP(status);
```

HPSETCCODE

NM callable only.

Sets the condition code for the calling process.

```
                I32V
HPSETCCODE(ccodevalue);
```

HPSETDUMP

NM callable only.

Arms the system debugger call from a process abort.

```
              I32   CA
HPSETDUMP(status,cmdstr);
```

HPSORTEND

NM callable only.

Releases the SORT/XL work area and ends the sorting operation.

```
              I32   I32A
HPSORTEND(status,statistics);
```

HPSORTERRORMESS

NM callable only.

Retrieves an error message if a fatal error occurs in SORT/XL.

```
                    I32    CA    I32
HPSORTERRORMESS(status,message,length);
```

HPSORTINIT

NM callable only.

Initializes the SORT/XL subsystem.

```
              I32    I32A      I32A       I32       I32V    I32V    I32V
HPSORTINIT(status,inputfiles,outputfiles,outputoption,reclength,numrecs,numkeys,
         I32A  CA     PROC      PROC      I32A    I32V    I32A
         keys,altseq,keycompare,errorproc,statistics,memsize,charseq);
```

HPSORTINPUT

NM callable only.

Passes records, one at a time, to SORT/XL.

```
                I32  CA   I32V
   HPSORTINPUT(status,buffer,length);
```

HPSORTOUTPUT

NM callable only.

Retrieves records, one at a time, from SORT/XL program.

```
                 I32   CA   I32
   HPSORTOUTPUT(status,buffer,length);
```

HPSORTSTAT

NM callable only.

Prints the SORT/XL statistics on $STDLIST.

```
               I32   I32A
   HPSORTSTAT(status,statistics);
```

HPSORTTITLE

NM callable only.

Prints the version number and title information for SORT/XL on $STDLIST and prints the date and time produced by the DATELINE intrinsic.

```
                I32
   HPSORTTITLE(status);
```

HPSWITCHTOCM

NM callable only.

Makes native mode (NM) to compatibility mode (CM) mixed-mode procedure calls possible.

```
              REC  I32V   I32V     RECA  I32V  RECV   I16   I32
HPSWITCHTOCM(proc,method,numparms,parms,fretlen,fretval,condcode,status);
```

HPSWTONMNAME

CM callable only.

Allows CM user programs, user libraries, and system code to invoke NM procedures as follows:

- Convert CM references in an argument list to virtual NM addresses.

- Change the execution mode.
- Invoke the NM procedure specified by the CM caller.

```
                CA      I16V     CA    I16V   I16V    I16    I16     I16V
HPSWTONMNAME(procname,proclen,libname,liblen,nparms,arglist,argdesc,functype);
```

HPSWTONMPLABEL

CM callable only.

Allows CM user programs, user libraries, and system code to invoke NM procedures as follows:

- Convert CM references in the argument list to virtual NM addresses.
- Change the execution mode.
- Invoke the NM procedure specified by the CM caller.

```
                  U32V I16V    I16     I16     I16V
HPSWTONMPLABEL(proc,nparms,arglist,argdesc,functype):
```

HPUNLOADCMPROCEDURE

NM callable only.

Unloads a target CM procedure whose plabel is obtained through the HPLOADCMPROCEDURE intrinsic.

```
                         CA       U8V   I32
HPUNLOADCMPROCEDURE(procname,library,status);
```

INITUSLF

NM and CM callable.

Initializes a USL file to the empty state. A USL contains CM object code and is meaningful only in the CM program development process.

```
   I16               I16V   I16A
uslferror:=INITUSLF(uslfnum,record);
```

IODONTWAIT

NM and CM callable.

Initiates completion operations for an I/O request.

```
 I16               I16V   UDS  I16   U16
fnum:=IODONTWAIT(filenum,buffer,length,cstation)
```

IOWAIT

NM and CM callable.

Initiates completion operations for an I/O request.

```
 I16          I16V    UDS  I16   U16
fnum:=IOWAIT(filenum,buffer,length,cstation);
```

JOBINFO

NM and CM callable.

Provides access to job and session information.

```
          I16V  I32    U16A      I16V    *     I16
JOBINFO(jsind,jsnum,jsstatus)[,itemnum,item,itemerror] [...];
```

Up to five *itemnum*/*item*/*itemerror* triples can be specified.

KILL

NM and CM callable.

Deletes a child process of the calling process and all of its descendants. Process handling (PH) capability is required.

```
     I16V
KILL(pin);
```

LOADPROC

NM and CM callable.

Dynamically loads a compatibility mode (CM) segmented library (SL) procedure and any external procedures it has referenced.

```
 I16                CA      I16V   I16
idnum:=LOADPROC(procname,library,plabel);
```

LOCKGLORIN

NM and CM callable.

Locks a global resource identification number (RIN). Multiple RIN (MR) capability is required to lock more than one global RIN simultaneously.

```
            I16V    U16       CA
LOCKGLORIN(rinnum,lockflag,rinpassword);
```

LOCKLOCRIN

NM and CM callable.

Locks a local resource identification number (RIN).

```
            I16V    U16
LOCKLOCRIN(rinnum,lockflag);
```

LOCRINOWNER

NM and CM callable.

Determines process identification number (PIN) of the process that locked a local resource identification number (RIN).

```
I16              I16V
pin:=LOCRINOWNER(rinnum);
```

LOGINFO

NM and CM callable.

Provides information about an opened user logging file (whole file set). User logging (LG) or system supervisor (OP) capability is required.

```
            I32V   I16        I16V   *
LOGINFO (index,logstatus [,itemnum,item] [...]);
```

Up to four *itemnum/item* pairs can be specified.

LOGSTATUS

NM and CM callable.

Provides information about a currently opened user logging file. User logging (LG) or system supervisor (OP) capability is required.

```
             I32   U16A    I16
LOGSTATUS(index,loginfo,logstatus);
```

MAIL

NM and CM callable.

Determines the status of the mailbox used by its parent or child. Process handling (PH) capability is required.

```
    U16         I16V  I16
mailstatus:=MAIL(pin,length);
```

MERGEEND

NM and CM callable.

Restores the data stack to its original state and ends the merging operation.

```
MERGEEND;
```

MERGEERRORMESS

NM and CM callable.

Retrieves a message if a fatal error occurs during the MERGE/XL operation and converts MERGEINIT error code values into ASCII strings.

```
                  I16V    CA     I16
MERGEERRORMESS(errorcode,message,length);
```

MERGEINIT

NM and CM callable.

Initializes the MERGE/XL subsystem and the merging of two or more sorted files.

```
             I16A    PROC      I16A       PROC     I16V   I16V
MERGEINIT(inputfiles,preprocessor,outputfiles,postprocessor,keysonly,numkeys,
I16A I16A   PROC     PROC    I16A   I16    I16      I16       I16A
keys,altseq,keycompare,errorproc,statistics,failure,errorparm,spaceallocation,charseq);
```

MERGEOUTPUT

NM and CM callable.

Provides an alternative method of specifying how records are output from the MERGE program.

```
                CA   I16
MERGEOUTPUT(record,length);
```

MERGESTAT

NM and CM callable.

Prints the MERGE program statistics on $STDLIST.

```
             I16A
MERGESTAT(statistics);
```

MERGETITLE

NM and CM callable.

Prints the version number and title of the merge segment on $STDLIST and prints the date and time produced by the DATELINE intrinsic.

```
MERGETITLE;
```

MYCOMMAND

NM and CM callable.

Parses (delineates and defines) parameters for a user-defined command image.

```
  I16                 CA       CA       I16V      I16    I32A
entrynum:=MYCOMMAND(cmdimage,delimiters,maxparms,numparms,params,
                    CA      @*
               dictionar,definition);
```

NLAPPEND

NM and CM callable.

Appends a language ID number to a file name that allows an application to designate which language-dependent file to use.

```
               CA       I16V    U16A
NLAPPEND(formaldesig,langnum,error);
```

NLCOLLATE

NM and CM callable.

Collates two character strings according to the specified language collating sequence and determines a lexical ordering.

```
              CA      CA      I16V       116    I16V   U16A  U16A
NLCOLLATE(buffer1,buffer2,bufferlength,result,langnum,error,collseq);
```

NLCONVCLOCK

NM and CM callable.

Converts the time format from a character string to numeric value; checks the input string using the formatting template returned by *itemnum*=3 of the NLINFO intrinsic, then converts the time to the general time format returned by the CLOCK intrinsic.

```
I32                 CA      I16V      I16V   U16A
time:=NLCONVCLOCK(buffer,bufferlength,langnum,error);
```

NLCONVCUSTDATE

NM and CM callable.

Converts the custom date format from a character string to a numeric value; checks the input string by using the formatting template returned by item 2 of the NLINFO intrinsic, then converts the date to the general date format as returned by the CALENDAR intrinsic.

```
 U16                  CA       I16V      I16V   U16A
date:=NLCONVCUSTDATE(buffer,bufferlength,langnum,error);
```

NLCONVNUM

NM and CM callable.

Converts native language numbers with native decimal and thousands separators to an ASCII number with NATIVE-3000 decimal and thousands separators. Optionally, the decimal and thousands separators can be removed.

```
             I16V    CA       I16V      CA       I16V   U16V
NLCONVNUM(langnum,instring,inlength,outstring,outlength,error,
          U16V   U16V     U16V          O-V
        numspec,fmtmask,decimals);
```

NLFINDSTR

NM and CM callable.

Searches **string1** for **string2**, and returns an integer value indicating the offset in **string1** where **string2** was found.

```
 I16              I16V    CA     I16V     CA    I16V   U16A  U16A
offset:=NLFINDSTR(langnum,string1,length1,string2,length2,error,charset);
```

NLFMTCALENDAR

NM and CM callable.

Formats the date according to language-dependent templates. The formatting is done according to the template returned by *itemnum*= 1 of the NLINFO intrinsic.

```
               U16V   CA     I16V   U16A
NLFMTCALENDAR(date,buffer,langnum,error);
```

NLFMTCLOCK

NM and CM callable.

Formats the time of day, in the specified language, obtained with the CLOCK intrinsic.

```
              I32V    CA     I16V     U16A
NLFMTCLOCK(time,buffer,langnum,error);
```

NLFMTCUSTDATE

NM and CM callable.

Formats the general date format returned by the CALENDAR intrinsic into the custom date format for a native language. A custom date is an abbreviated format such as 10/1/82 or 82.10.1. The formatting is done according to the template returned by *itemnum*= 2 of the NLINFO intrinsic.

```
                U16V    CA     I16V    U16A
NLFMTCUSTDATE(date,buffer,langnum,error);
```

NLFMTDATE

NM and CM callable.

Formats the date and time according to language-dependent templates returned by *itemnum*s 1 and 3 of the NLINFO intrinsic.

```
             U16V I32V    CA     I16V    U16A
NLFMTDATE(date,time,buffer,langnum,error);
```

NLFMTLONGCAL

NM and CM callable.

Formats the supplied date according to the long calendar format. The formatting is done according to the template returned by NLINFO *itemnum*=30.

```
                LV     BA      IV      LA
NLFMTLONGCAL(date,string,langnum,error)
```

NLFMTNUM

NM and CM callable.

Converts a string containing an ASCII number (can include NATIVE-3000 decimal separator (.), thousands separator (,), and currency symbol/name ($)) to a language-specific format using the decimal separator, thousands separator, and currency symbol/name defined for the native language.

```
               I16V      CA      I16V       CA       I16V     U16A
NLFMTNUM(langnum,instring,inlength,outstring,outlength,error,
           U16A     U16V    I16V          O-V
         numspec,fmtmask,decimals)
```

NLGETLANG

NM and CM callable.

Returns a language ID number that characterizes the current user, data, or system. Hewlett-Packard subsystems and application programs use NLGETLANG for automatic configuration.

```
   I16                 I16V    U16A
langnum:=NLGETLANG(langtype,error);
```

NLINFO

NM and CM callable.

Returns language-dependent information. The type of information that can be obtained includes:

- Calendar format
- Date and time format
- Currency
- Collating
- Translation
- Character set

```
            I16V    *   I16   U16A
NLINFO(itemnum,item,langnum,error);
```

NLJUDGE

NM and CM callable.

Judges whether a character is a 1 byte or 2 byte Asian character.

```
   I16V             I16V     CA         I16V       CA  U16A   U16A
n2bytes:=NLJUDGE(langnum,instring,stringlength,flags,error,charset);
```

NLKEYCOMPARE

NM and CM callable.

Compares two strings of different length (for use with KSAM generic key searching).

```
                 CA       I16V   CA  I16V    I16      I16V    U16A  U16A
NLKEYCOMPARE(generickey,length1,key,length2,result,langnum,error,collseq);
```

NLNUMSPEC

NM and CM callable.

Returns the information needed for formatting and converting numbers. It combines several calls to NLINFO to simplify the use of native language formatting. By calling NLNUMSPEC once, and passing the obtained information to NLFMTNUM and NLCONVNUM, implicit calls to NLNUMSPEC from NLFMTNUM and NLCONVNUM are avoided and performance is improved.

```
              I16V    U16A  U16A
NLNUMSPEC(langnum,string,error);
```

NLREPCHAR

NM and CM callable.

Replaces all nondisplayable control characters in the string with the replacement character. Nondisplayable characters are those with attribute 3 (undefined graphic character) or 5 (control code), as returned by *itemnum*=12 of the NLINFO intrinsic.

```
              CA       CA        I16V        CV        I16V   U16A  U16A
NLREPCHAR(inbuffer,outbuffer,bufferlength,replacechar,langnum,error,charset);
```

NLSCANMOVE

NM and CM callable.

Scans and moves character strings according to character attributes. This function is handled in a language-dependent manner.

```
  I16                   CA       CA     U16V    I16V      I16V    U16A
numchar:=NLSCANMOVE(inbuffer,outbuffer,flags,bufferlength,langnum,error,
           U16A   CA
        charset,shiftinfo);
```

NLSUBSTR

NM and CM callable.

Extracts **movelength** bytes from the **instring** to the **outstring**.

```
           CA      I16V      CA      I16        I16V       16V
NLSUBSTR(instring,inlength,outstring,outlength,startposition,movelength,
         I16V    I16V  U16A  U16A
       langnum,flags,error,charset);
```

NLSWITCHBUF

NM and CM callable.

Converts a string of characters from phonetic order to screen order or from screen order to phonetic order.

```
              I16V     CA       CA        I16V        U16V        U16A
NLSWITCHBUF(langnum,instring,outstring,stringlength,left-to-right,error);
```

NLTRANSLATE

NM and CM callable.

Translates a string of characters from EBCDIC-to-ASCII or ASCII-to-EBCDIC using the appropriate native language table.

```
              I16V       CA       CA       I16V       I16V   U16A
NLTRANSLATE(transcode,inbuffer,outbuffer,bufferlength,langnum,error,

    CA
transtable);
```

OPENLOG

NM and CM callable.

Provides access to the user logging facility. User logging (LG) or system supervisor (OP) capability is required.

```
          I32   CA   CA   I16    I16
OPENLOG(index,logid,pass,mode,logstatus);
```

PAUSE

NM and CM callable.

Suspends the calling process for a specified number of seconds.

```
         32R
PAUSE(interval);
```

PRINT

NM and CM callable.

Prints character string on job/session listing device.

```
       CA     I16V    I16V
PRINT(message,length,controlcode);
```

PRINTFILEINFO

NM and CM callable.

Prints a file or directory information display on the job/session list device.

```
                I16V
PRINTFILEINFO(filenum);
```

PRINTOP

NM and CM callable.

Prints a character string on the system console.

```
           CA     I16V     I16V
PRINTOP(message,length,controlcode);
```

PRINTOPREPLY

NM and CM callable.

Prints a character string on the system console and solicits a reply.

```
 I16                   CA    I16V  I16V  CA     I16V
length:=PRINTOPREPLY(message,length,zero,reply,maxlength);
```

PROCINFO

NM and CM callable.

Provides access to process information.

```
           I16    I16 I16V    I16V   *
PROCINFO(error1,error2,pin[,itemnum,item] [...]);
```

Up to six *itemnum/item* pairs can be specified.

PROCTIME

NM and CM callable.

Returns the accumulated CPU time for a process.

```
   I32
time:=PROCTIME;
```

PUTJCW

NM and CM callable.

Assigns the value of a particular job control word (JCW) in the job control word table.

```
            CA      U16      I16
PUTJCW(jcwname,jcwvalue,jcwstatus);
```

QUIT

NM and CM callable.

Aborts the calling process.

```
       I16V
QUIT(num);
```

QUITPROG

NM and CM callable.

Aborts the entire user process structure.

```
           I16V
QUITPROG(num);
```

READ

NM and CM callable.

Reads an ASCII string from $STDIN into an array.

```
    I16            CA    I16V
length:=READ(message,msglength);
```

READX

NM and CM callable.

Reads an ASCII string from $STDINX into an array.

```
    I16            CA     I16V
length:=READX(message,msglength);
```

RECEIVEMAIL

NM and CM callable.

Receives mail from another process. Process handling (PH) capability is required.

```
     U16                 I16V  UDS     U16V
mailstatus:=RECEIVEMAIL(pin,location,waitflag);
```

RESETCONTROL

NM and CM callable.

Reenables the subsystem break trap which allows a process to accept other subsystem break signals.

```
RESETCONTROL;
```

RESETDUMP

NM and CM callable.

Disables the abort stack analysis facility. Only the current process is affected.

```
RESETDUMP;
```

SEARCH

NM and CM callable.

Searches a specially-formatted array for a specified entry or name.

```
     I16             CA   I16V     CA          @*
entrynum:=SEARCH(buffer,length,dictionary,definition);
```

SENDMAIL

NM and CM callable.

Sends mail to another process. Process handling (PH) capability is required.

```
     U16             I16V I16V    UDS     U16V
mailstatus:=SENDMAIL(pin,length,location,waitflag);
```

SETDUMP

NM and CM callable.

Arms a call to the system debugger from a process abort.

```
         I16V
SETDUMP(flags);
```

SETJCW

NM and CM callable.

Sets bits in the system job control word (JCW).

```
        U16V
SETJCW(jcword);
```

SORTEND

NM and CM callable.

Closes the scratch file and restores the data stack to its original state.

```
SORTEND;
```

SORTERRORMESS

NM and CM callable.

Retrieves and prints a message if a fatal error occurs during the SORT program.

```
                    I16V    CA     I16
SORTERRORMESS(errorcode,message,length);
```

SORTINIT

NM and CM callable.

Initiates the SORT program.

```
             I16A   I16A     I16V     I16V   I32V    I16V  I16A
SORTINIT( inputfiles,outputfiles,outputoption,reclength,numrecs,numkeys,keys,
 I16A    PROC      PROC  I16A   I16     I16         I16     I16A
altseq,keycompare,errorproc,statistics,failure,errorparm,spaceallocation,charseq);
```

SORTINPUT

NM and CM callable.

Provides an alternative method of specifying how records are supplied to the SORT program.

```
              CA   I16V
SORTINPUT(record,length);
```

SORTOUTPUT

NM and CM callable.

Provides an alternative method of specifying how records are output from the SORT program.

```
              CA   I16
SORTOUTPUT(record,length);
```

SORTSTAT

NM and CM callable.

Prints the SORT program statistics on $STDLIST. Call SORTSTAT after you have called the SORTEND intrinsic.

```
             I16A
SORTSTAT(statistics);
```

SORTTITLE

NM and CM callable.

Prints the version number and title of the SORTLIB segment on $STDLIST.

```
SORTTITLE;
```

STACKDUMP

NM and CM callable.

Calls the system debugger to send a stack trace to $STDLIST or to the file specified in the *formaldesig* parameter. Control then returns to the calling procedure.

(NM and CM)

```
              CA        I16V  I16V I32
STACKDUMP(formaldesig,idnumber,flags,selec);
```

(CM: SPL language only)

```
              CA        I16V  I16V I32
STACKDUMP'(formaldesig,idnumber,flags,selec);
```

STARTSESS

NM and CM callable.

Initiates a session on the specified terminal. Programmatic sessions (PS) capability is required.

```
           I16V     CA     I16  I32    I16A
STARTSESS(ldev,logonstring,jsid,jsnum,jsstatus);
```

SUSPEND

NM and CM callable.

Suspends a process. Process handling (PH) capability is required.

```
          U16V I16V
SUSPEND(allow,rin);
```

SWITCHDB

CM callable only.

Switches the DB register pointer. Privileged mode (PM) capability is required.

```
    U16      O-P    U16V
logindex:=SWITCHDB(index)
```

TERMINATE

NM and CM callable.

Releases all resources held by the process and its descendants are released. All remaining files, opened by the process and its descendants, are closed and assigned the same disposition they had when opened.

```
TERMINATE;
```

TIMER

NM and CM callable.

Returns system timer information.

```
 I32
count:=TIMER;
```

UNLOADPROC

NM and CM callable.

Dynamically unloads a compatibility mode (CM) segmented library (SL) procedure.

```
            I16V
UNLOADPROC(procid);
```

UNLOCKGLORIN

NM and CM callable.

Unlocks a global resource identification number (RIN) that was locked with the LOCKGLORIN intrinsic.

```
              I16V
UNLOCKGLORIN(rinnum);
```

UNLOCKLOCRIN

NM and CM callable.

Unlocks a local resource identification number (RIN) that was locked by the LOCKLOCRIN intrinsic.

```
              I16V
UNLOCKLOCRIN(rinnum);
```

WHO

NM and CM callable.

Returns the access mode and attributes of the user calling the intrinsic.

```
      U16   I32      I32     CA      CA        CA      CA     U16
WHO(mode,capability,localattr,username,groupname,acctname,homename,term);
```

WRITELOG

NM and CM callable.

Writes database and subsystem file records to the user logging file. User logging (LG) or system supervisor (OP) capability is required.

```
          I32  U16A  I16   I16    I16
WRITELOG(index,data,length,mode,logstatus);
```

XARITRAP

NM and CM callable.

Arms or disarms the user-written arithmetic trap handling procedure.

```
          I*V  I32V    I32      I32
XARITRAP(mask,plabel,oldmask,oldplabel);
```

XCONTRAP

NM and CM callable.

Arms or disarms user-written subsystem break trap handling procedure.

```
          I*V     I*
XCONTRAP(plabel,oldplabel);
```

XLIBTRAP

NM and CM callable.

Enables or disables a user-written software library trap handling procedure.

```
          I*V     I*
XLIBTRAP(plabel,oldplabel);
```

XSYSTRAP

NM and CM callable.

Enables or disables a user-written system trap handling procedure.

```
          I*V     I*
XSYSTRAP(plabel,oldplabel);
```

ZSIZE

NM and CM callable.

Alters current DB to Z area of the compatibility mode (CM) stack.

```
   I16         I16V
newsize:=ZSIZE(size);
```

4 FCOPY Commands

FCOPY commands

Description of all the FCOPY commands.

To Initiate FCOPY

```
RUN FCOPY.PUB.SYS

FROM[=fromfile
     =tofile
     =*
     =];TO[=(dfile,kfile)
           =(tofile)
           =tofile
           =*
           =][;functionlist]
```

Syntax of FCOPY Functions

```
[;NOUSERLABELS][;CCTL
                ;NOCCTL][;NEW]
[;{CLEAR
   KANA}[;HEX
         ;OCTAL
         ;HEXO][;NORECNUM][;TITLE=title]]
[;CHAR[;HEX
       ;OCTAL
       ;HEXO][;NORECNUM][;TITLE=title][;LANG= language ]]
[;{HEX
   OCTAL
   HEXO}[;CHAR
         ;CLEAR
         ;KANA][;NORECNUM][;TITLE=title]]
[;DEBLOCK=logical-record-length]
[;{EBCDICIN
   EBCDICOUT}[={field
                (field[;field[;...]])} [,EXCLUDE][;LANG=language]]]
[;{BCDICIN
   BCDICOUT
   EBCDIKIN
   EBCDIKOUT}[={field
                (field[;field[;...]])}[,EXCLUDE]]]
[;FILES={number-of-files
         ALL}]
[;IGNERR[=number-of-errors]][;COMPARE[=number-of-errors]]
[;SKIPEOF=[{+
            -}from-eofs
           from-file-number][,{+
                                -}to-eofs
                              ,to-file-number]]
[;SUBSET=["characterstring"[,column][,EXCLUDE]
          #patternlist#[,column][,EXCLUDE]
          (range[;range][;...])]]
[;NOKSAM]      [;KEY[=character-location]]
[;UPSHIFT[;LANG=language]][;VERIFY[=number-of-errors]]
```

FCOPY Functions

BCDICIN/BCDICOUT

BCDICIN translates from BCDIC to ASCII. BCDICOUT translates from ASCII to BCDIC.

```
;{BCDICIN
  BCDICOUT}[={field
              (field[;field[;...]])} [,EXCLUDE]]
FROM=FILE1;TO=FILE2;BCDICIN=(1,5;10:30),EXCLUDE
```

CCTL/NOCCTL

CCTL designates the first character of each record in the *fromfile* as a carriage control character in the *tofile*; NOCCTL specifies that the first character of each record in the *fromfile* is not to be used as a carriage control character in the *tofile*.

```
;{CCTL
  NOCCTL}
FILE BETA;NOCCTL
```

CHAR

Displays the contents of a file, record by record, in the form of character symbols in ASCII code.

```
;CHAR[;HEX
      ;HEXO
      ;OCTAL] [;NORECNUM] [;TITLE=title][;LANG=language]
FCOPY FROM=DISPL;TO=;OCTAL;CHAR
```

CLEAR

Displays the contents of a file, record by record, in the form of character symbols for all codes in the file.

```
;CLEAR[;HEX
       ;HEXO
       ;OCTAL][;NORECNUM][;TITLE=title]
FCOPY FROM=DISPL;TO=;OCTAL;CLEAR
```

COMPARE

Compares the contents of the *fromfile* with the contents of the *tofile*, record by record, without changing either file.

```
;COMPARE[=number-of-errors]
FROM=FILEA;TO=DUP1;COMPARE
```

COPYACD

Copies the access control definition (ACD) associated with a file when the file is being copied.

```
;COPYACD
FROM=SOURCEF;TO=TARGETF;COPYACD
```

COPYACD applies only to MPE V Delta 4 and subsequent releases and not to MPE/iX.

DEBLOCK

Removes a record from the blocked status.

```
;DEBLOCK=logical-record-length
FILE TAPEBYTE;REC=-790,1,U,ASCII
FROM=*TAPEBYTE;TO=DISC1;DEBLOCK=-79
```

EBCDICIN/EBCDICOUT

EBCDICIN translates from EBCDIC to the character code specified in the translation table of the language you select. EBCDICOUT translates from the character code specified in the translation table of the language you select to EBCDIC. When you do not specify a language, EBCDICIN translates from EBCDIC to ASCII, and EBCDOUT translates from ASCII to EBCDIC.

```
;{EBCDICIN
  EBCDICOUT}
[={field
   (field[;field[;...]])}[,EXCLUDE][;LANG=language]]
FROM=*TAPE;TO=DISC1;EBCDICIN=3:7,14:27
```

EBCDIKIN/EBCDIKOUT

EBCDIKIN translates from EBCDIK (IBM Standard) to JIS (Japanese Industrial Standard). EBCDIKOUT translates from JIS to EBCDIK.

```
;{EBCDIKIN
  EBCDIKOUT}[={field
              (field[;field[;...]])}[,EXCLUDE]]
FROM=FILE1;TO=FILE2;EBCDIKIN=3:6,EXCLUDE
```

FILES

Copies multiple files from unlabeled magnetic tapes, serial disks, and cartridge tapes. FCOPY copies only one file if you do not use the FILES function.

```
;FILES={number-of-files
        ALL}
FROM=*TAPEA;TO=*TAPEB;FILES=3;SUBSET=11:25
```

HEX

Displays the contents of a file, record by record, in the form of character code numbers in hexadecimal form.

```
;HEX[;CHAR
     ;CLEAR
     ;KANA][;NORECNUM][;TITLE=title]
FROM=TEXT3;TO=*LP;HEX;CHAR;
TITLE="TITLE LINE FOR CHAR/HEX DISPLAY EXAMPLE"
```

HEXO

Displays the contents of a file, record by record, in the form of character code numbers, the data in hexadecimal form, and the record number in octal form.

```
;HEXO[;CHAR
      ;CLEAR
      ;KANA][;NORECNUM][;TITLE=title]
FROM=TEXT3;TO=*LP;HEXO;CHAR;
TITLE="TITLE LINE FOR CHAR/HEX DISPLAY EXAMPLE"
```

IGNERR

Bypasses errors in a magnetic tape *fromfile* and reports each ignored error.

```
;IGNERR[=number-of-errors]
FROM=*TAPE;TP=FILE3;SUBSET;IGNERR=5
```

KANA

Displays the contents of a file, record by record, in the form of JIS character symbols. KANA displays symbols not represented by characters in JIS code as decimal points.

```
;KANA[;HEX
      ;HEXO
      ;OCTAL][;NORECNUM][;TITLE=title]
; KANA;OCTAL;TITLE="KANA symbols in OCTAL"
```

KEY

Chooses a key sequence in which to copy KSAM flies. The KEY function works only with KSAM *fromfiles*.

```
;KEY [=character-location]
FROM=KSAM;TO=ALPHA;KEY=21
```

NEW

Creates a new permanent disk file as the *tofile*.

```
;NEW
FROM=OLDSTUFF;TO=NEWFILE;NEW
```

NOKSAM

Copies the data file of a KSAM file into another, non-KSAM file.

```
;NOKSAM
FROM=KSAMFILE;TO=FILEX;NOKSAM;NOUSERLABELS
```

NOUSERLABELS

NOUSERLABELS lets you omit user labels when copying from a tape or disk file to another file.

```
;NOUSERLABELS
FROM=*TAPEA;TO=DISC;NOUSERLABELS
```

OCTAL

Lets you display the contents of a file, record by record, in the form of character code numbers in octal form.

```
;OCTAL[;CHAR
      ;CLEAR
      ;KANA ][;NORECNUM][;TITLE=title]
FROM=TEXT3;TO=*LP;OCTAL
```

SKIPEOF

SKIPEOF instructs FCOPY to skip end-of-file markers on a serial storage device, in order to position the device at the desired file before copying. SKIPEOF is not applicable to labeled tapes.

```
;SKIPEOF=[{+
           -}from-eofs
          from-file-number][,{+
                              -}to-eofs
                              ,to-file-number]
FROM=*THISTAPE;TO=*THATTAPE;SKIPEOF=4,5
```

SUBSET

SUBSET lets you copy only a specific portion (subset) of a file. You can define the subset in one of two ways, either as all records with a certain character string or numeric pattern beginning in a specific column, or as a set of continuous records.

```
;SUBSET[="characterstring"[,column][,EXCLUDE]
        =#patternlist#[,column][,EXCLUDE]
        =(range[;range][;...])]
FROM=MASTER;TO=MEN;SUBSET="MALE",17
```

UPSHIFT

UPSHIFT converts lowercase Roman alphabetic characters to uppercase as part of the copying operation.

```
;UPSHIFT[;LANG=language]
FROM=LOWER;TO=UPPER;UPSHIFT
```

VERIFY

VERIFY compares the contents of the *tofile* with the contents of the *fromfile*, record by record, immediately after a copy operation.

```
;VERIFY[=number-of-errors]
FROM=OLDDISC;TO=COPY;VERIFY
```

5 SORT-MERGE/XL Commands

Description of SORT-MERGE/XL Commands

To Initiate SORT

```
RUN SORT.PUB.SYS
```

ALTSEQ

The ALTSEQ command defines a collating sequence other than the standard ASCII or EBCDIC format. The ALTSEQ command must be preceded by a DATA command. It is effective only if the keys are of *type* BYTE and if the input data is ASCII.

```
A[LTSEQ]modspec1[,modspec2]...[, modspecN]

[EACH]leftspec{=
              <blank>
              WITH} rightspec
or
MERGE   leftspec   {WITH
                   <blank>
                   =  } rightspec
```

To specify *leftspec* and *rightspec* use the following form:

```
{string
 num byte
 range string }
```

DATA

Specifies the type of the input data (either ASCII or EBCDIC) and the basic collating sequence to be used in the particular SORT/XL (or MERGE/XL) operation. The collating sequence may be altered, if desired, by using the ALTSEQ command.

```
DATA [IS] {A[SCII]
           E[BCDIC]} [ , ] SEQ[UENCE] [IS]{ A[SCII]
                                            E[BCDIC]}
```

END

Specifies the conclusion of SORT-MERGE/XL parameters. It also starts the sort or merge operation specified.

```
E[ND]
```

EXIT

Terminates the operation of SORT/XL or MERGE/XL and exits the subsystem.

```
EX[IT]
```

INPUT (SORT/XL)

Within the SORT/XL subsystem, the INPUT command specifies the input file(s) to be sorted. Refer to the MERGE/XL INPUT command for information on how to use the command within that subsystem.

```
I[NPUT]  {$STDIN [ X ]
          *
          fname
          (filename1,filename2,...filenameN)}[,#records][,rec size]
```

KEY

Specifies the location of the key data items in a file's records which are to be sorted or merged.

```
K[EY] keyspec1 [; keyspec2]...[; keyspecN]
```

keyspec — A group of parameters used to specify a key data item to be sorted or merged. The syntax of the *keyspec* parameters follows:

```
position, length [,type][,DESC]
```

LANGUAGE

Defines the native language whose collating sequence is to be used to sort keys of type CHARACTER.

```
L[ANGUAGE][IS]   {langnum
                  <blank>
                  langname}
```

OUTPUT (SORT/XL)

Designates and creates the output file which is to receive the sorted records. Refer to the MERGE/XL OUTPUT command for information on how to use the command within that subsystem.

```
O[UTPUT]  {*
           $STDLIST
           filename }[, NUM][, KEY]
```

RESET

The RESET command is used to correct errors made in the specification of keys. When entered, it nullifies all existing KEY commands.

```
RESET
```

SHOW

Displays the collating sequence or the translation table.

```
SH[OW]  {S[EQUENCE][,O[FFLINE
         T[ABLE][,O[FFLINE
         <blank>
         NOS[EQUENCE]
         NOT[ABLE]            }
```

VERIFY

Displays information on the input and output files, key descriptions, and the various options in effect during a SORT/XL or MERGE/XL operation to the file LIST.

```
V[ERIFY]
```

:(MPE Command)

The : is entered preceding MPE commands within SORT/XL or MERGE/XL, for example, for entering file equations.

```
: [MPE command]
```

:EOD

The :EOD command is not truly a command. It terminates the list of input records to MERGE/XL when * (for $STDIN) is the input file.

```
:EOD
```

To Initiate MERGE

```
RUN MERGE.PUB.SYS
```

ALTSEQ

The ALTSEQ command defines a collating sequence other than the standard ASCII or EBCDIC format. The ALTSEQ command must be preceded by a DATA command. It is effective only if the keys are of *type* BYTE and if the input data is ASCII. (Refer to Appendix B of the *Sort-Merge/XL General User's Guide* for information on ASCII and EBCDIC character set values.)

A[LTSEQ] *modspec1*[, *modspec2*]..[, *modspecN*]

[EACH]*leftspec* { =
<blank>
WITH} *rightspec*
or
MERGE *leftspec* {WITH
<blank>
= } *rightspec*

To specify *leftspec* and *rightspec* use the following form:

{*string*
num byte
range string }

DATA

Specifies the type of the input data (either ASCII or EBCDIC) and the basic collating sequence to be used in the particular SORT/XL (or MERGE/XL) operation. The collating sequence may be altered, if desired, by using the ALTSEQ command.

```
DATA [IS] {A[SCII]
           E[BCDIC]}  [ , ] SEQ[UENCE] [IS] {A[SCII]
                                             E[BCDIC]}
```

END

Specifies the conclusion of SORT-MERGE/XL parameters. It also starts the sort or merge operation specified.

```
E[ND]
```

EXIT

Terminates the operation of SORT/XL or MERGE/XL and exits the subsystem.

```
EX[IT]
```

INPUT (MERGE/XL)

Within the MERGE/XL subsystem, the INPUT command specifies the sorted files to be merged. Refer to the SORT/XL INPUT command for information on how to use the command within that subsystem.

```
I[NPUT] {filename1,filename2}[,filename3]...[,filenameN]
```

KEY

Specifies the location of the key data items in a file's records which are to be sorted or merged.

```
K[EY] keyspec1 [; keyspec2]...[; keyspecN]
```

keyspec — A group of parameters used to specify a key data item to be sorted or merged. The syntax of the *keyspec* parameters follows:

```
position, length [,type][,DESC]
```

LANGUAGE

Defines the native language whose collating sequence is to be used to sort keys of type CHARACTER.

```
L[ANGUAGE][IS] {langnum
                <blank>
                langname}
```

OUTPUT (MERGE/XL)

The OUTPUT command is used to designate and create the output file, which is to receive the merged records. Refer to the SORT/XL OUTPUT command for information on how to use the command within that subsystem.

```
O[UTPUT]  {filename
           <blank>
           $STDLIST}[,num records][, KEY]
```

RESET

The RESET command is used to correct errors made in the specification of keys. When entered, it nullifies all existing KEY commands.

```
RESET
```

SHOW

Displays the collating sequence or the translation table.

```
SH[OW]  {S[EQUENCE][,O[FFLINE
         T[ABLE][,O[FFLINE
         <blank>
         NOS[EQUENCE]
         NOT[ABLE]            }
```

VERIFY

Displays information on the input and output files, key descriptions, and the various options in effect during a SORT/XL or MERGE/XL operation to the file LIST.

```
V[ERIFY]
```

:(MPE Command)

The : is entered preceding MPE commands within SORT/XL or MERGE/XL.

```
: [MPE command]
```

:EOD

The :EOD command is not truly a command. It terminates the list of input records to SORT/XL when * (for $STDIN) is the input file.

```
:EOD
```

6 System Debug

System Debug provides a family of low-level assembly language debuggers for MPE/iX:

Debugging your system

- Debug
- Dump Analysis Tool (DAT)
- Standalone Analysis Tool (SAT)

This chapter presents short descriptions of System Debug commands, window commands, standard functions, and environment variables. Refer to the *System Debug Reference Manual* for additional details on System Debug commands and functions described in this chapter.

System Debug Command Descriptions

This section presents short descriptions of System Debug commands. Commands that are inappropriate in either DAT or Debug are identified as "DAT only" or "Debug only". In addition, commands that require privileged mode (PM) capability are identified.

:

The CI command - Access to the MPE/iX command interpreter (CI).

```
: [ command ]
```

=

The calculator command. Calculates the value of an expression and displays the result in the specified base.

```
= expression [base]
```

ABORT

Aborts/terminates the current System Debug process.

```
ABORT
```

ALIAS

Defines an alias (alternative) name for a command or macro.

```
ALIAS name command
```

ALIASD[EL]

Deletes the specified alias(es).

```
ALIASD[EL] pattern [group]
```

ALIASINIT

Restores the predefined aliases, in case they have been deleted.

```
ALIASINIT
```

ALIASL[IST]

Lists the currently defined aliases.

```
ALIAS[LIST] [pattern] [group]
```

B (break)

Debug only. Privileged Mode: BA, BAX, BS.

Break. Sets a breakpoint.

```
B    logaddr [:pin|@] [count] [loud] [cmdlist]          Program
BG   logaddr [:pin|@] [count] [loud] [cmdlist]          Group library
BP   logaddr [:pin|@] [count] [loud] [cmdlist]          Account library
BLG  logaddr [:pin|@] [count] [loud] [cmdlist]          Logon group lib
BLP  logaddr [:pin|@] [count] [loud] [cmdlist]          Logon account lib
BS   logaddr [:pin|@] [count] [loud] [cmdlist]          System library

BU fname logaddr [:pin|@] [count] [loud] [cmdlist]     User library

BV   virtaddr [:pin|@] [count] [loud] [cmdlist]          Virtual address

BA   cmabsaddr [:pin|@] [count] [loud] [cmdlist]        Absolute CST
BAX  cmabsaddr [:pin|@] [count] [loud] [cmdlist]        Absolute CSTX
```

BD

Debug only.

Breakpoint delete. Deletes a breakpoint entry specified by index number.

```
BD [number | @ [: pin | @] ]
```

BL

Debug only.

Breakpoint list. Lists breakpoint entries, specified by index number.

```
BL [number | @ [: pin | @] ]
```

CLOSEDUMP

DAT only.

Closes a dump file.

```
CLOSEDUMP
```

CM

Enters compatibility mode (cmdat/cmdebug). See the NM command.

```
CM
```

CMDL[IST]

Command list. Displays a list of the valid commands for System Debug.

```
CMDL[IST] [pattern] [group] [options]
```

CMG

Privileged Mode

Displays values in the CMGLOBALS record for a process.

```
CMG  [pin]
```

C[ONTINUE]

Continues/resumes execution of user program.

```
C[ONTINUE]
C[ONTINUE] [IGNORE]
C[ONTINUE] [NOIGNORE]
```

D (display)

Privileged Mode: DA, DCS, DCA, DZ, DSEC.

Displays the contents of the specified address.

```
DA    offset    [count] [base] [recw] [bytew]         ABS relative
DD    dst.off   [count] [base] [recw] [bytew]          CM data segment
DDB   offset    [count] [base] [recw] [bytew]         DB relative
DS    offset    [count] [base] [recw] [bytew]         S relative
DQ    offset    [count] [base] [recw] [bytew]         Q relative

DC    logaddr   [count] [base] [recw] [bytew]        Program file
DCG   logaddr   [count] [base] [recw] [bytew]        Group library
DCP   logaddr   [count] [base] [recw] [bytew]        Account library
DCLG  logaddr   [count] [base] [recw] [bytew]        Logon group lib
DCLP  logaddr   [count] [base] [recw] [bytew]        Logon account lib
DCS   logaddr   [count] [base] [recw] [bytew]        System library
DCU   fname logaddr [count] [base] [recw] [bytew]  User library

DCA   cmabsaddr [count] [base] [recw] [bytew]     Absolute CST
DCAX  cmabsaddr [count] [base] [recw] [bytew]     Absolute CSTX

DV    virtaddr   [count] [base] [recw] [bytew]       Virtual
DZ    realaddr [count] [base] [recw] [bytew]        Real memory

DSEC  ldev.off [count] [base] [recw] [bytew]         Secondary store
```

DATAB

Debug only. Privileged Mode.

Sets a data breakpoint.

```
DATAB virtaddr [:pin|@] [byte_count] [count] [loudness] [cmdlist]
```

DATABD

Debug only. Privileged Mode.

Deletes a data breakpoint entry specified by index number.

```
DATABD [number | @ [: pin | @] ]
```

DATABL

Debug only. Privileged Mode.

Lists data breakpoint entries, specified by index number.

```
DATABL [number | @ [: pin | @] ]
```

DEBUG

Debug only. Privileged Mode.

DEBUG command access to DEBUG XL.

```
DEBUG
```

DELETE*xxx*

Delete various items. These are predefined aliases for other commands.

```
DELETEB        alias for  BD
DELETEALIAS    alias for  ALIASD
DELETEERR      alias for  ERRD
DELETEMAC      alias for  MACD
DELETEVAR      alias for  VARD
```

DEMO

Privileged Mode.

Adds/deletes/lists terminals used for demonstrating System Debug.

```
DEMO
DEMO LIST
DEMO ADD    ldevs
DEMO DELETE ldevs
```

DIS

Disassembles a single NM or CM assembly instruction, based on the current mode.

```
DIS nmword [virtaddr]
DIS cmword1 [cmword2] [cmlogaddr]
```

DO

Reexecutes a command from the command stack.

```
DO [cmd_string ]
DO [history_index]
```

DPIB

DAT Privileged Mode.

Display data from the process identification block (PIB) for a process.

```
DPIB  [pin]
```

DPTREE

DAT Privileged Mode.

Prints out the process tree starting at the given PIN.

```
DPTREE [pin]
```

DR

Displays contents of the CM or NM registers.

```
DR [cm_register] [base]
DR [nm_register] [base]
```

DUMPINFO

DAT Privileged Mode.

Displays dump file information.

```
DUMPINFO [options]
```

ENV

Assigns a new value to one of the predefined environment variables.

```
ENV var_name [=] var_value
```

ENVL[IST]

Displays the current values for environmental variables.

```
ENVL[IST] [pattern] [group] [options]
```

ERR

Pushes a user error message onto the error command stack.

```
ERR errmsg
```

ERRD[EL]

Deletes all errors on the error stack (reset the stack).

```
ERRD[EL]
```

ERRL[IST]

Error list. Lists the most recent error(s) on the error stack.

```
ERRL[IST] [ALL]
```

E[XIT]

Exits/resumes execution of user program.

```
E[XIT]      Same as CONTINUE (in Debug)
E[XIT]      Exit program       (in DAT)
```

F (format)

Formats a specified data structure.

```
FT                  path  ft_options

FV  virtaddr  path  fv_options
```

F (freeze)

Debug only. Privileged Mode.

Freezes a code segment, data segment, or virtual address (range) in memory.

```
FC      logaddr     [bytelength]          Program file
FCG     logaddr     [bytelength]          Group library
FCP     logaddr     [bytelength]          Account library
FCLG    logaddr     [bytelength]          Logon group library
FCLP    logaddr     [bytelength]          Logon account library
FCS     logaddr     [bytelength]          System library
FCU   fname logaddr     [bytelength]    User library

FCA     cmabsaddr                         CM absolute CST
FCAX    cmabsaddr                         CM absolute CST

FDA     dstoff                             CM data segment

FVA     virtaddr     [bytelength]        Virtual address
```

FINDPROC

Debug Privileged Mode.

Dynamically loads a specified NM procedure from any NM library.

```
FINDPROC procedurename library_file  [ [NO]IGNORECASE]
```

FOREACH

Each time a FOREACH command is executed, *name* is set to the next expression value in *value_list* prior to the execution of *cmdlist*. Execution ends when there are no more expression values in the *value_list*.

```
FOREACH   name   value_list         command

FOREACH    name  value_list      { cmdlist }
```

FPMAP

Reinitializes CM FPMAP symbolic procedure name access.

```
FPMAP
```

FUNCL[IST]

Function list. Displays information about the predefined functions.

```
FUNCL[IST] [pattern]  [group]  [options]
```

GETDUMP

DAT Privileged Mode.

Reads in a dump tape and creates a dump file.

```
GETDUMP file [ ldevlist ]
GETDUMP file [ DIR ]
```

H[ELP]

Displays online help messages for System Debug.

```
H[ELP] [topic] [options]
```

HIST[ORY]

Displays the history command stack.

```
HIST[ORY] option
```

IF

If *condition* evaluates to TRUE, then execute all commands in *cmdlist,* else execute all commands in *cmdlist2*.

```
IF condition THEN command

IF condition THEN { cmdlist }

IF condition THEN   command1  ELSE command2

IF condition THEN { cmdlist } ELSE command2

IF condition THEN  command1   ELSE { cmdlist2 }

IF condition THEN { cmdlist } ELSE { cmdlist2 }
```

IGNORE

Protects the next command (list) from error bailout.

```
IGNORE option
```

INIT*xx*

Privileged Mode.

Initialize registers from a specified location.

```
INITNM virtaddr [ISM |PIMREAL |PIMVIRTUAL]
INITCM virtaddr [ISM |PIMREAL |PIMVIRTUAL]

INITNM TCB
INITCM TCB | CMG | REGS
```

KILL

Debug only

Privileged Mode

Issues a request to process management to kill the specified process.

```
KILL pin
```

LEV

Sets the current environment to the specified stack level in the stack markers.

```
LEV [number]
LEV [number] [interrupt_level]
```

LIST

Controls the recording of input and output to a list file.

```
LIST

LIST  [filename]

LIST  [ON ]
LIST  [OFF]

LIST  [CLOSE]
```

LISTREDO

Displays the history command stack.

```
LISTREDO              alias for HIST[ORY]
```

LOADINFO

Debug only

Lists information about the currently loaded program and libraries.

```
LOADINFO
```

LOADPROC

Debug only.

Dynamically loads a specified CM procedure from a logically specified CM library selector.

```
LOADPROC procedurename libselect
```

LOC

Defines a local variable within a macro body.

```
LOC  var_name [:var_type] [=] var_value
```

LOCL[IST]

Lists the local variables that are defined with a macro.

```
LOCL[IST] [pattern]
```

LOG

Controls the recording of user input to the logfile.

```
LOG

LOG [filename]

LOG [ON   ]
LOG [OFF  ]
LOG [CLOSE]
```

M (modify)

Debug only. Privileged Mode: MA, MD, MCS, MZ, MSEC.

Modifies the contents of the specified number of words at the specified address.

```
MA    offset    [count] [base] [newvalue(s)]        ABS relative
MD    dst.off   [count] [base] [newvalue(s)]        Data segment
MDB   offset    [count] [base] [newvalue(s)]        DB relative
MS    offset    [count] [base] [newvalue(s)]        S relative
MQ    offset    [count] [base] [newvalue(s)]        Q relative

MC    logaddr   [count] [base] [newvalue(s)]         Program file
MCG   logaddr   [count] [base] [newvalue(s)]         Group library
MCP   logaddr   [count] [base] [newvalue(s)]         Account library
MCLG  logaddr   [count] [base] [newvalue(s)]         Logon group
MCLP  logaddr   [count] [base] [newvalue(s)]         Logon account
MCS   logaddr   [count] [base] [newvalue(s)]         System library
MCU   fname logaddr   [count] [base] [newvalue(s)] User library
MCA   cmabsaddr   [count] [base] [newvalue(s)]     Absolute CST
MCAX  cmabsaddr   [count] [base] [newvalue(s)]     Absolute CSTX

MV    virtaddr    [count] [base] [newvalue(s)]       Virtual
MZ    realaddr    [count] [base] [newvalue(s)]      Real memory

MSEC  ldev.off    [count] [base] [newvalue(s)]       Secondary storem
```

MAC[RO]

Defines a macro.

```
MAC[RO] name {body}
MAC[RO] name [ (parameters) ] {body}
MAC[RO] name [ (parameters) ] [options] {body}
```

MACD[EL]

Macro delete. Deletes the specified macro definition(s).

```
MACD[EL] pattern
```

MACECHO

Controls the "echoing" of each macro command line prior to its execution.

```
MACECHO pattern [level]
```

MACL[IST]

Macro list. Lists the specified macro definition(s).

```
MACL[IST] [pattern]  [options]
```

MACREF

Resets the reference count to zero for the specified macro(s).

```
MACREF pattern
```

MACTRACE

Controls the "tracing" of macro execution.

```
MACTRACE pattern [level]
```

MAP

Opens a file and maps it into a usable virtual address space.

```
MAP filename [option]
```

MAPL[IST]

Lists the specified file(s) that have been opened with the MAP command.

```
MAPL[IST] [pattern]
```

MODD

DAT Privileged Mode.

Modification delete. Deletes a modification entry specified by index number.

```
MODD [index
      @]
```

MODL

DAT only.

Modification list. Lists current dump modifications.

```
MODL [index
      @]
```

MPEXL

Privileged Mode.

Displays information about the files which were used to build the operating system SOM portion of the NL.Pub.SYS for MPE/iX.

```
MPEXL [fileset] [optionstring] [outputfile]
```

MPSW

Privileged Mode.

Modifies the NM processor status word (PSW). Exercise a bit of care with this command.

```
MPSW bit_string
```

MR

Modifies the contents of the specified CM or NM register.

```
MR cm_register [newvalue]
MR nm_register [newvalue]
```

NM

Enters native mode (nmdat / nmdebug). See the CM command.

```
NM
```

OPENDUMP

DAT Privileged Mode.

Opens a dump file.

```
OPENDUMP file
```

PAUSE

Pauses (puts to sleep) a process for the specified number of seconds.

```
PAUSE n
```

PIN

Privileged Mode.

Switches the process-specific pointers and registers to allow the examination of process related information.

```
PIN [pin] [ANYSTATE]
```

PROCLIST

Lists the specified NM symbols in the specified NM executable library.

```
PROCLIST [pattern] [lstfile] [lookup_id] [detail] [outputfile]
```

PURGEDUMP

DAT Privileged Mode.

Purges a dump file.

```
PURGEDUMP file
```

REDO

Reexecutes a command from the history command stack after optionally editing the command.

```
REDO [cmd_string ]
REDO [history_index]
```

REGLIST

Lists the registers into a file in USE file format.

```
REGLIST [filename]
```

RESTORE

Restores macros or variables from a file that was previously created by the STORE command.

```
RESTORE MACROS    filename
RESTORE VARIABLES filename
```

RET[URN]

Exits from a macro, optionally returning a specified value.

```
RET[URN] [value]
```

SET

Set new values for a select subset of all user configurable options.

```
SET

SET [ O[CT] | %
      D[EC] | #
      H[EX] | $ ] [ IN
                    OUT ]

SET [ CRON
      CROFF ]

SET [ MOREON
      MOREOFF ]

SET [ DEF[AULT] ]
```

SET*xxx*

The SET*xxx* commands are predefined aliases for other commands.

```
SETALIAS    alias for  ALIAS
SETENV      alias for  ENV
SETERR      alias for  ERR
SETLOC      alias for  LOC
SETMAC      alias for  MAC
SETVAR      alias for  VAR
```

SHOW*xxx*

The SHOW*xxx* commands are predefined aliases for other commands.

```
SHOWALIAS    alias for  ALIASL
SHOWB        alias for  BL
SHOWCMD      alias for  CMDL
SHOWDATAB    alias for  DATABL
SHOWENV      alias for  ENVL
SHOWERR      alias for  ERRL
SHOWFUNC     alias for  FUNCL
SHOWLOC      alias for  LOCL
SHOWMAC      alias for  MACL
SHOWMAP      alias for  MAPL
```

```
SHOWSET      alias for  SET
SHOWSYM      alias for  SYML
SHOWVAR      alias for  VARL
```

S, SS

Single steps.

```
S[S]  [num_instrs] [ L[OUD] | Q[UIET] ]
```

STORE

Stores the currently defined macros or variables to a file.

```
STORE MACROS     filename
STORE VARIABLES filename
```

SYMCLOSE

Closes a symbolic data type file that was opened with the SYMOPEN command.

```
SYMCLOSE symname
```

SYMF[ILES]

Lists all open symbolic data type files and their symbolic names.

```
SYMF[ILES]
```

SYMINFO

Lists information/dump data for an opened symbolic data type file.

```
SYMINFO [symname] [option] [offset] [length]
```

SYML[IST]

Lists information for the specified symbol name in an opened symbolic data type file.

```
SYML[IST] [pattern] [symname] [option]
```

SYMOPEN

Opens a symbolic data type file and sets up pointers to the symbolic debug records.

```
SYMOPEN filename [symname]
```

SYMPREP

Prepares a program file containing symbolic debug information to be used by the symbolic formatter/symbolic access facility. Files modified through the use of this command are referred to as symbolic data type files.

```
SYMPREP filename
```

T (translate)

Privileged Mode: TCA, TCS.

Translates the specified CM address to a virtual address.

```
TA    offset                    ABS - Bank0
TD    dst.off                    Data segment
TDB   offset                    DB relative
TS    offset                    S relative
TQ    offset                    Q relative

TC    cmlogaddr                 Program file
TCG   cmlogaddr                 Group library
TCP   cmlogaddr                 Account library
TCLG  cmlogaddr                 Logon group library
TCLP  cmlogaddr                 Logon account library
TCS   cmlogaddr                 System library

TCA   cmabsaddr                 Absolute CST
TCAX  cmabsaddr                 Absolute CSTX
```

TERM

Debug only.

Controls the synchronization of several debug processes on a single terminal.

```
TERM
TERM LIST
TERM NEXT
```

TR[ACE]

Displays a stack trace.

```
TR[ACE] [level] [options]
```

TRAP

Debug only.

Arms/disarms/lists various traps that are monitored by Debug.

```
TRAP [LIST]
TRAP [trap-name] [option]
```

UF

Debug Privileged Mode.

Unfreezes a code segment, data segment, or virtual address (range) in memory.

```
UFC    logaddr   [bytelength]          Program file
UFCG   logaddr   [bytelength]          Group library
UFCP   logaddr   [bytelength]          Account library
UFCLG  logaddr                          Logon group library
UFCLP  logaddr                          Logon account library
UFCS   logaddr   [bytelength]          System library
UFCU   fname logaddr [bytelength]     User library

UFCA  cmabsaddr                        Absolute CST
UFCAX cmabsaddr                         Absolute CSTX
```

```
UFDA   dst.off                          CM data segment
UFVA   virtaddr [bytelength]            Virtual address
```

UNMAP

Closes (unmaps) a file that was opened by the MAP command.

```
UNMAP index
```

UPD

Update the windows.

```
UPD
```

USE

System Debug commands can be executed from a file with the USE command.

```
USE
USE  [filename]  [count]
USENEXT  count
USE [CLOSE][ALL | @]
```

VAR

Defines a user-defined variable.

```
VAR  var_name [:var_type] [=] var_value
```

VARD[EL]

Variable delete. Deletes the specified user-defined variable(s).

```
VARD[EL]  pattern
```

VARL[IST]

Variable list. Lists the value(s) for the specified user-defined variable(s).

```
VARL[IST]  [pattern]
```

W (write)

Writes a list of values, with optional formatting, to output.

```
W     valuelist
WL    valuelist
WP    valuelist

WCOL column
WPAGE
```

WHELP

Displays online help messages for the window commands.

```
WHELP
```

WHILE

While *condition* evaluates to TRUE, executes all commands in *cmdlist*.

```
WHILE condition DO cmdlist
```

XL

Utilizes symbol information in a local library/program file.

```
XL localfile space_id [loaded-fname]
```

XLD

Closes files opened with the XL command.

```
XLD localfile
```

XLL

Lists all of the files that have been opened with the XL command.

```
XLL
```

Window Commands

This section presents short descriptions of System Debug window commands.

RED

Redraws the entire screen display of windows.

```
RED
```

UW*m*

Allocates a named user window at the specified address. The command name specifies which type of window to define. User windows are displayed within the group window.

```
UWA    offset      [name]    Absolute memory relative (ABS)
UWDB   offset      [name]    DB relative
UWS    offset      [name]    S relative
UWQ    offset      [name]    Q relative

UWD    dst.off     [name]    Data segment and offset

UWCA   cmabsaddr   [name]   Code (CST) segment and offset
UWCAX  cmabsaddr   [name]   Code (CSTX) segment and offset

UWV    virtaddr    [name]    Virtual address
UWZ    realaddr   [name]    Real address
```

WDEF

Window defaults. Resets the default window sizes.

```
WDEF
```

WGRP

Changes to the specified group of user-defined windows.

```
WGRP  [group_number]
```

WOFF

Windows OFF. Turns off the windows.

```
WOFF
```

WON

Windows ON. Turns on the windows. If windows are already on, redraws them.

```
WON
```

*w*B

Window back. Scrolls the specified window backwards.

```
PB   [amount]                      Program, current mode
CMPB [amount]                      CM program
NMPB [amount]                      NM program

QB   [amount]                      CM frame,  Q relative
SB   [amount]                      CM stack,  S relative

GB   [amount]                      Group window
UB   [amount] [win_number]        User window

VB   [amount] [win_number]        Virtual window
ZB   [amount]                       Real memory window
LB   [amount]                       LDEV window
TXB  [amount] [win_number]        Text window
```

*w*C

Window current. Marks the specified window as the current window. Many user window (U), text window (TX), and virtual window (V) commands operate on the current window.

```
UC  [win_number]
VC  [win_number]
TXC [win_number]
```

*w*D

Window disable.

```
RD                         CM registers
GRD                        NM general registers
SRD                        NM special registers

PD                         Program, current mode
CMPD                       CM program
NMPD                       NM program
QD                         CM frame,  Q relative
SD                         CM stack,  S relative

GD                         Group window

UD    [win_number]        User  window
```

```
VD    [win_number]        Virtual window
ZD                          Real memory window
LD                          LDEV window
TXD   [win_number]        Text window
```

*w*E

Window enable.

```
RE                          CM registers
GRE                         NM general registers
SRE                         NM special registers

PE                          Program, current mode
CMPE                        CM program
NMPE                        NM program
QE                          CM Frame,  Q relative
SE                          CM Stack,  S relative

GE                          Group window

UE    [win_number]        User window

VE    [win_number]        Virtual window
ZE                          Real memory window
LE                          LDEV window
TXE   [win_number]        Text window
```

*w*F

Window forward. Scrolls the specified window forward.

```
PF   [amount]                     Program current mode
CMPF [amount]                     CM program
NMPF [amount]                     NM program

QF   [amount]                     CM frame,  Q relative
SF   [amount]                     CM stack,  S relative

GF   [amount]                     Group window
UF   [amount] [win_number]      User window

VF   [amount] [win_number]      Virtual window
ZF   [amount]                     Real memory window
LF   [amount]                     LDEV window
TXF  [amount] [win_number]      Text window
```

*w*H

Window home. Returns a window to its original location.

```
RH                                  CM registers window
GRH                                 NM general registers window
SRH                                 NM special registers window

PH                                  Program window, current mode
CMPH                                CM program window
NMPH                                NM program window
QH                                  CM frame window - Q relative
SH                                  CM stack window - S relative

GH                                  Group window
UH  [win_number]                  User window

VH  [virtaddr]   [win_number]     Virtual window
ZH  [realaddr]                     Real memory window
LH  [ldev.off]                     LDEV window
TXH [win_number]                  Text window
```

*w*I

Window information. Prints information about the indicated windows. This command is defined for the virtual (V) and text (TX) windows.

```
VI  [win_number]
TXI [win_number]
```

*w*J

Window jump. Jumps window to the specified address.

```
PJ     [logaddr]                    Program file
PJG    [logaddr]                    Group library
PJP    [logaddr]                    Account library
PJLG   [logaddr]                    Logon group library
PJLP   [logaddr]                    Logon account library
PJS    [logaddr]                    System library
PJU    [fname logaddr]              User library
PJV    [virtaddr]                    Any virtual address
PJA    [absaddr]                    Absolute CST
PJAX   [absaddr]                    Absolute CSTX

CMPJ   [logaddr]                    Program file
CMPJG  [logaddr]                    Group library
CMPJP  [logaddr]                    Account library
CMPJLG [logaddr]                    Logon group library
CMPJLP [logaddr]                    Logon account library
CMPJS  [logaddr]                    System library
CMPJA  [absaddr]                    Absolute CST
CMPJAX [absaddr]                    Absolute CSTX

NMPJ   [logaddr]                    Program file
NMPJG  [logaddr]                    Group library
NMPJP  [logaddr]                    Account library
NMPJLG [logaddr]                    Logon group library
NMPJLP [logaddr]                    Logon account library
NMPJS  [logaddr]                    System library
NMPJU  [fname logaddr]              User library

QJ     [dst.off]                     CM Frame,  Q relative
SJ     [dst.off]                     CM Stack,  S relative

VJ     [virtaddr]   [win_number]    Virtual window
ZJ     [realaddr]                    Real memory window
LJ     [Ldev.off]                   LDEV window
TXJ    [record_number]              Text window
```

*w*K

Window kill.

```
RK                          CM registers
GRK                         NM general registers
SRK                         NM special registers

PK                          Program, current mode
CMPK                        CM program
NMPK                        NM program
QK                          CM frame,  Q relative
SK                          CM stack,  S relative

GK                          Group window
UK     [win_number]        User window

VK     [win_number]        Virtual window
ZK                          Real memory window
LK                          LDEV window
TXK    [win_number]        Text window
```

*w*L

Window lines. Sets the number of lines in a window.

```
RL   [numlines]                      CM registers
GRL  [numlines]                      NM general registers
SRL  [numlines]                      NM special registers

PL   [numlines]                      Program, current mode
CMPL [numlines]                      CM program
NMPL [numlines]                      NM program
QL   [numlines]                      CM frame,  Q relative
SL   [numlines]                      CM stack,  S relative

GL   [numlines]                      Group window
UL   [numlines] [win_number]        User window

VL   [numlines] [win_number]        Virtual window
ZL   [numlines]                      Real memory window
LL   [numlines]                      LDEV window
TXL  [numlines] [win_number]        Text window
```

*w*M

Window mode. Changes the mode for the Q or S window.

```
QM  [addressmode] [signed]
SM  [addressmode] [signed]
```

*w*N

Renames a virtual window or a user-defined window.

```
UN   [name] [win_number]    User window
VN   [name] [win_number]    Virtual window
```

*w*R

Sets the radix (output base) for the specified window.

```
RR   base                        CM registers

PR   base                        Program, current mode
CMPR base                        CM program
NMPR base                        NM program
QR   base                        CM frame,  Q relative
SR   base                        CM stack,  S relative

GR   base                        Group window
UR   base [win_number]          User window

VR   base [win_number]          Virtual window
ZR   base                        Real memory window
LR   base                        Ldev window
```

*w*S

Window shift. Shifts a window to the left or right. This command is defined for text windows (TX).

```
TXS  [ amount ] [win_number]
```

*w*W

Defines (enables) new windows.

```
VW  virtaddr  [name]        Virtual window
ZW  realaddr                  Real Memory
LW  Ldev.offf                 LDEV (Secondary Storage) window
TXW filename                  Text window

UWm                            User window (see UWm command)
```

System Debug Function Specifications

This section presents short descriptions of the standard functions defined in System Debug. All functions are callable from both DAT and Debug.

func abstolog

Converts an CM absolute code address (ACPTR) to a CM logical code (LCPTR) address.

```
abstolog (cmabsaddr)
```

Formal Declaration

```
abstolog:lcptr (cmabsaddr:acptr)
```

func asc

Evaluates an expression and converts the result to an ASCII string.

```
asc (value [formatspec])
```

Formal Declaration

```
asc:str (value:any [formatspec:str = ''])
```

func ascc

Coerces an expression into a string value.

```
ascc (value)
```

Formal Declaration

```
ascc:str (value:any)
```

func bin

Converts a string expression to return a binary value.

```
bin (strexp)
```

Formal Declaration

```
bin:any (strexp:str)
```

func bitd

Bit deposit. Deposits a value into a specified range of bits.

```
bitd (value position length target)
```

Formal Declaration

```
bitd:any (value:any position:s16 length:u16 target:any)
```

func bitx

Bit extract. Extracts a range of bits from an expression.

```
bitx (source position length)
```

Formal Declaration

```
bitx:any (source:any position:s16 length:u16)
```

func bool

Coerces an expression into a Boolean value.

```
bool (value)
```

Formal Declaration

```
bool:bool (value:any)
```

func bound

Checks for an existing definition of an operand and returns its definition type.

```
bound (operand)
```

Formal Declaration

```
bound:str (operand:str)
```

func btow

Byte to word. Converts a CM DB-relative byte address to a CM DB-relative word address.

```
btow (byteaddress [splitstack])
```

Formal Declaration

```
btow:I16 (byteaddress:I16 [splitstack:bool=FALSE])
```

func cisetvar

Sets a new value for the specified CI (MPE/iX Command Interpreter) variable.

```
cisetvar (civarname newvalue)
```

Formal Declaration

```
cisetvar:bool (civarname:str newvalue:any)
```

func civar

Returns the current value of a CI (MPE/iX Command Interpreter) variable.

```
civar (civarname [stropt])
```

Formal Declaration

```
civar:any (civarname:str [stropt:str="NOEV"])
```

func cmaddr

Converts a CM procedure name (or primary/secondary entry point) to a CM logical code address.

```
cmaddr (procname [lib])
```

Formal Declaration

```
cmaddr:lcptr (procname:str [lib:str=''])
```

func cmbpaddr

Returns the address corresponding to the indicated CM breakpoint index.

```
cmbpaddr (bpindex [pin])
```

Formal Declaration

```
cmbpaddr:lcptr (bpindex:u16 [pin:s16=0])
```

func cmbpindex

Returns the CM breakpoint index associated with the indicated CM code address.

```
cmbpindex (cmaddr [pin])
```

This function accepts the address (either logical or absolute) of an existing CM breakpoint and returns the logical index number associated with that breakpoint. The default action is to look for breakpoints set by the current PIN. Breakpoint indices for other PINs (including the global PIN) may be retrieved by utilizing the optional *pin* parameter.

Formal Declaration

```
cmbpindex:u16 (cmaddr:cptr [pin:s16=0])
```

func cmbpinstr

Returns the original CM instruction at a specified CM code address where a CM breakpoint has been set.

```
cmbpinstr (cmaddr [pin])
```

Formal Declaration

```
cmbpinstr:s16 (cmaddr:cptr [pin:s16=0])
```

func cmentry

Returns the CM (primary) entry point address of the CM procedure containing the specified CM logical code address.

```
cmentry (cmlogaddr)
```

Formal Declaration

```
cmentry:lcptr (cmlogaddr:lcptr)
```

func cmg

Returns the virtual address (SPTR) of a process's CMGLOBALS record.

```
cmg (pin)
```

Formal Declaration

```
cmg:sptr (pin:u16)
```

func cmnode

Returns the address of the closest CM node point corresponding to the specified CM logical code address.

```
cmnode (cmlogaddr [node])
```

Formal Declaration

```
cmnode:lcptr (cmlogaddr:lcptr [node:str="PREV"])
```

func cmproc

Returns the CM procedure name and offset corresponding to a CM logical code address.

```
cmproc (cmlogaddr)
```

Formal Declaration

```
cmproc:str (cmlogaddr:lcptr)
```

func cmproclen

Returns the length of the CM procedure which contains the specified CM logical code address.

```
cmproclen (cmlogaddr)
```

Formal Declaration

```
cmproclen:u16 (cmlogaddr:lcptr)
```

func cmseg

Returns the CM segment name for the specified CM logical code address.

```
cmseg (cmlogaddr)
```

Formal Declaration

```
cmseg:str (cmlogaddr:lcptr)
```

func cmstackbase

Returns the starting virtual address of a process's compatibility mode stack.

```
cmstackbase (pin)
```

Formal Declaration

```
cmstackbase:lptr (pin:u16)
```

func cmstackdst

Returns the DST number for a process's compatibility mode stack.

```
cmstackdst (pin)
```

Formal Declaration

```
cmstackdst:u16 (pin:u16)
```

func cmstacklimit

Returns the virtual address for the limit of a process's compatibility mode stack.

```
cmstacklimit (pin)
```

Formal Declaration

```
cmstacklimit:lptr (pin:u16)
```

func cmstart

Returns the starting point of the procedure containing the indicated CM logical code address.

```
cmstart (cmlogaddr)
```

Formal Declaration

```
cmstart:lcptr (cmlogaddr:lcptr)
```

func cmtonmnode

Returns the address of the closest NM node point corresponding to the specified CM logical code address.

```
cmtonmnode (cmlogaddr [node])
```

Formal Declaration

```
cmtonmnode:trans (cmlogaddr:lcptr [node:str=PREV])
```

func cmva

Returns the virtual address of a specified CM code address.

```
cmva (cmaddr [pin])
```

Formal Declaration

```
cmva:lptr (cmaddr:cptr [pin:u16 = 0])
```

func cst

Coerces an expression into a CST absolute code pointer (ACPTR).

```
cst (value)
```

Formal Declaration

```
cst:cst (value:any)
```

func cstx

Coerces an expression into a CSTX absolute code pointer (ACPTR).

```
cstx (value)
```

Formal Declaration

```
cstx:cstx (value:any)
```

func dstva

Converts a CM data segment address to a virtual address.

```
dsvta (dstoff)
```

Formal Declaration

```
dstva:lptr (dstoff:lptr)
```

func errmsg

Returns an error message string, based on error number and an optional subsystem number.

```
errmsg (errnum [subsys])
```

Formal Declaration

```
errmsg:str (errnum:s16 [subsys:u16=$a9])
```

func grp

Coerces an expression into a GRP logical code pointer (LCPTR).

```
grp (value)
```

Formal Declaration

```
grp:grp (value:any)
```

func hash

Hashes a virtual address into a hash table (real) offset.

```
hash (virtaddr)
```

Formal Declaration

```
hash:s32 (virtaddr:ptr)
```

func lgrp

Coerces an expression into a LGRP logical code pointer (LCPTR).

```
lgrp (value)
```

Formal Declaration

```
lgrp:lgrp (value:any)
```

func logtoabs

Logical to absolute. Converts a CM logical code address (LCPTR) into a CM absolute code address (ACPTR).

```
logtoabs (cmlogaddr)
```

Formal Declaration

```
logtoabs:acptr (cmlogaddr:lcptr)
```

func lptr

Coerces an expression into a long pointer.

```
lptr (value)
```

Formal Declaration

```
lptr:lptr (value:any)
```

func lpub

Coerces an expression into a LPUB logical code pointer (LCPTR).

```
lpub (value)
```

Formal Declaration

```
lpub:lpub (value:any)
```

func ltolog

Long to logical. Converts a long pointer into a NM logical code address (LCPTR).

```
ltolog (longptr)
```

Formal Declaration

```
ltolog:lcptr (longptr:lptr)
```

func ltos

Long to short. Converts a virtual address to a short pointer.

```
ltos (virtaddr)
```

Formal Declaration

```
ltos:sptr (virtaddr:ptr)
```

func macbody

Returns a string that is the macro body for the specified macro name.

```
macbody (macroname)
```

Formal Declaration

```
macbody:str (macroname:str)
```

func mapindex

Returns the map index number of the specified file name which has been previously mapped into virtual space with the MAP command.

```
mapindex (filename)
```

Formal Declaration

```
pindex:u16 (filename:str)
```

func mapsize

Returns the size in bytes of the specified mapped file.

```
mapsize (filename)
```

Formal Declaration

```
mapsize:u32 (filename:str)
```

func mapva

Returns the virtual address of the specified mapped file.

```
mapva (filename)
```

Formal Declaration

```
mapva:lptr (filename:str)
```

func nmaddr

Returns the virtual address of the specified NM procedure/data path.

```
nmaddr (path [lookupid])
```

Formal Declaration

```
nmaddr:long (path:str [lookupid:str="PROCEDURE"])
```

func nmbpaddr

Returns the address corresponding to the indicated NM breakpoint index.

```
%nmbpaddr (bpindex [pin])
```

Formal Declaration

```
nmbpaddr:lptr (bpindex:u32 [pin:s16=0])
```

func nmbpindex

Returns the NM breakpoint index for the NM breakpoint that has been set at the specified NM code address.

```
mbpindex (virtaddr [pin])
```

Formal Declaration

```
nmbpindex:u32 (virtaddr:ptr [pin:s16=0])
```

func nmbpinstr

Returns the original NM instruction at a specified NM code address where a NM breakpoint has been set.

```
nmbpinstr (virtaddr[pin])
```

Formal Declaration

```
nmbpinstr:s32 (virtaddr:ptr [pin:s16=0])
```

func nmcall

Dynamically calls a procedure/function passing up to four parameters.

```
nmcall (path) [parm1] [parm2] [parm3] [parm4]
```

Formal Declaration

```
nmcall:s32 (path:str [parm1:sptr=0][parm2:sptr=0]
[parm3:sptr=0] [parm4:sptr=0]
```

func nmentry

Returns the entry point of the NM procedure containing the indicated address.

```
nmentry (virtaddr)
```

Formal Declaration

```
nmentry:lptr (virtaddr:ptr)
```

func nmfile

Returns the file name corresponding to the indicated NM (code) address.

```
nmfile (virtaddr [length])
```

Formal Declaration

```
nmfile:str (virtaddr:ptr [length:u16=$20])
```

func nmmod

Returns the NM module name corresponding to the indicated address.

```
nmmod (virtaddr [length])
```

Formal Declaration

```
nmmod:str (virtaddr:ptr [length:u16=$20])
```

func nmnode

Returns the NM logical code address (TRANS) of the closest NM node point corresponding to the specified NM address.

```
nmnode (virtaddr [node])
```

Formal Declaration

```
nmnode:trans (virtaddr:ptr [node:str="PREV"])
```

func nmpath

Returns the full NM code path name corresponding to the indicated address.

```
nmpath (virtaddr [length])
```

Formal Declaration

```
nmpath:str (virtaddr:ptr [length:u16=$50])
```

func nmproc

Returns the NM procedure name and offset corresponding to the specified virtual address.

```
nmproc (virtaddr [length])
```

Formal Declaration

```
nmproc:str (virtaddr:ptr [length:u16=$40])
```

func nmstackbase

Returns the virtual address of the start of the process's NM stack.

```
nmstackbase (pin)
```

Formal Declaration

```
nmstackbase:lptr (pin:u16)
```

func nmstacklimit

Returns the virtual address of the limit of a process's NM stack.

```
nmstacklimit (pin)
```

Formal Declaration

```
nmstacklimit:lptr (pin:u16)
```

func nmtocmnode

Returns the CM logical code address of the closest CM node point corresponding to the specified NM address.

```
nmtocmnode (virtaddr [node])
```

Formal Declaration

```
nmtocmnode:lcptr (virtaddr:lptr [node:str="PREV"])
```

func off

Returns the offset portion of a virtual address.

```
off (virtaddr)
```

Formal Declaration

```
off:u32 (virtaddr:ptr)
```

func pcb

Returns the virtual address (SPTR) of a process's PCB (process control block).

```
pcb (pin)
```

Formal Declaration

```
pcb:sptr (pin:u16)
```

func pcbx

Returns the virtual address (SPTR) of a process's PCBX (process control block extension).

```
pcbx (pin)
```

Formal Declaration

```
pcbx:sptr (pin:u16)
```

func phystolog

Converts a CM physical segment number and mapping bit to a CM logical code address.

```
phystolog (physsegnum [mappingbit])
```

Formal Declaration

```
phystolog:lcptr (physsegnum:u16 [mappingbit:bool=FALSE])
```

func pib

Returns the virtual address (SPTR) of a process's process information block (PIB).

```
pib (pin)
```

Formal Declaration

```
pib:sptr (pin:u16)
```

func pibx

Returns the virtual address (SPTR) of a process's process information block extension (PIBX).

```
pibx: (pin)
```

Formal Declaration

```
pibx:sptr (pin:u16)
```

func prog

Coerce an expression into a PROG logical code pointer (LCPTR).

```
prog (value)
```

Formal Declaration

```
prog:prog (value:any)
```

func pstate

Returns the process state, for the specified PIN, as a string.

```
pstate (pin)
```

Formal Declaration

```
pstate:str (pin:u16)
```

func pub

Coerces an expression into a PUB logical code pointer (LCPTR).

```
pub (value)
```

Formal Declaration

```
pub:pub (value:any)
```

func rtov

Real to virtual. Converts a real address to a virtual address.

```
rtov (realaddr)
```

Formal Declaration

```
rtov:lptr (realaddr:u32)
```

func s16

Coerces an expression into a signed 16-bit value.

```
s16 (value)
```

Formal Declaration

```
s16:s16 (value:any)
```

func s32

Coerces an expression into a signed 32-bit value.

```
s32 (value)
```

Formal Declaration

```
s32:s32 (value:any)
```

func s64

Coerces an expression into a signed 64-bit value.

```
s64 (value)
```

Formal Declaration

```
s64:s64 (value:any)
```

func sid

Returns the space ID (SID) portion from a virtual address.

```
sid (virtaddr)
```

Formal Declaration

```
sid:u32 (virtaddr:ptr)
```

func sptr

Coerces an expression into a short pointer.

```
sptr (value)
```

Formal Declaration

```
sptr:sptr (value:any)
```

func stol

Short to long. Converts a virtual address to a long pointer.

```
stol (virtaddr)
```

Formal Declaration

```
stol:lptr (virtaddr:ptr)
```

func stolog

Short to logical. Converts a NM short pointer (SPTR) to a NM logical code address (LCPTR).

```
stolog (shortptr [logsel] [userfname])
```

Formal Declaration

```
stolog:lcptr (shortptr:sptr [logsel:str="PROG"] [userfname:str])
```

func str

Returns a substring of a source string.

```
str (source position length)
```

Formal Declaration

```
str:str (source:str position:u16 length:u16)
```

func strapp

String append. Returns the result of concatenating two strings.

```
strapp (source tail)
```

Formal Declaration

```
strapp:str (source:str tail:str)
```

func strdel

String delete. Returns a string with a substring deleted from the source string.

```
strdel (source position length)
```

Formal Declaration

```
strdel:str (source:str position:u16 length:u16)
```

func strdown

String downshift. Returns a string that is the result of downshifting all alphabetic characters in the source string.

```
strdown (source)
```

Formal Declaration

```
strdown:str (source:str)
```

func strextract

String extract. Returns a string (extracted) from the specified virtual address.

```
strextract (virtaddr [length])
```

Formal Declaration

```
strextract:str (virtaddr:ptr [length:u16=$4])
```

func strinput

Prompts on the input device for user input and returns the user input line as a string.

```
strinput (prompt)
```

Formal Declaration

```
strinput:str (prompt:str)
```

func strins

String insert. Returns a string after inserting another string into the source string.

```
strins (insert source position)
```

Formal Declaration

```
strins:str (insert:str source:str position:u16)
```

func strlen

String length. Returns the current size of a string.

```
strlen (source)
```

Formal Declaration

```
strlen:u32 (source:str)
```

func strltrim

String left trim. Deletes leading blanks from the source string.

```
strltrim (source)
```

Formal Declaration

```
strltrim:str (source:str)
```

func strmax

String maximum. Returns the (constant) maximum size of a string.

```
strmax (source)
```

Formal Declaration

```
strmax:u32 (source:str)
```

func strpos

String position. Returns the index of the first occurrence of one string in another.

```
strpos (source searchstring [position])
```

Formal Declaration

```
strpos:u32 (source:str searchstring:str [position:u32=1])
```

func strrpt

String repeat. Returns a string composed of repeated occurrences of a source string.

```
strrpt (source count)
```

Formal Declaration

```
strrpt:str (source:str count:u32)
```

func strrtrim

String right trim. Deletes trailing blanks from the source string.

```
strrtrim (source)
```

Formal Declaration

```
strrtrim:str (source:str)
```

func strup

String upshift. Returns a string which is the result of upshifting all alphabetic characters in the source string.

```
strup (source)
```

Formal Declaration

```
strup:str (source:str)
```

func strwrite

Returns a string which is the result of formatting one or more expressions in a manner equivalent to that of the W (WRITE) command.

```
strwrite (valuelist)
```

Formal Declaration

```
strwrite:str (valuelist:str)
```

func symaddr

Returns the bit- or byte-relative offset of a component specified through the path specification, relative to the outer structure.

```
symaddr (path [units])
```

Formal Declaration

```
symaddr:u32 (path:str [units:u16=8])
```

func symconst

Returns the value of a declared constant.

```
symconst (path)
```

Formal Declaration

```
symconst:any (path:str)
```

func syminset

Returns a Boolean value of TRUE if the set member specified by the member parameter is in the set specified by the virtual address and the path specification.

```
syminset (virtaddr path member)
```

Formal Declaration

```
syminset:bool (virtaddr:ptr path:str member:str)
```

func symlen

Returns the length of a data structure in bits or bytes.

```
symlen (path [units])
```

Formal Declaration

```
symlen:u32 (path:str [units:u32=$8])
```

func symtype

Returns the type of a component described by the path specification.

```
symtype  (path)
```

Formal Declaration

```
symtype:int (path:str)
```

func symval

Returns the value of a simple data type specified by a virtual address and a path.

```
symval (virtaddr path)
```

Formal Declaration

```
symval:any (virtaddr:ptr path:str)
```

func sys

Coerces an expression into a SYS logical code pointer (LCPTR).

```
sys (value)
```

Formal Declaration

```
sys:sys (value:any)
```

func tcb

Returns the real address of a process's TCB (task control block).

```
tcb (pin)
```

Formal Declaration

```
tcb:u32 (pin:u16)
```

func trans

Coerces an expression into a TRANS logical code pointer (LCPTR).

```
trans (value)
```

Formal Declaration

```
trans:trans (value:any)
```

func typeof

Returns the type of an evaluated expression as a string.

```
typeof (expr)
```

Formal Declaration

```
typeof:str (expr:any)
```

func u16

Coerces an expression into an unsigned 16-bit value.

```
u16 (value)
```

Formal Declaration

```
u16:u16 (value:any)
```

func u32

Coerces an expression into an unsigned 32-bit value.

```
u32 (value)
```

Formal Declaration

```
u32:u32 (value:any)
```

func user

Coerces an expression into a USER library logical code pointer (LCPTR).

```
user ([library] value)
```

Formal Declaration

```
user:user ([library:str=''] value:any)
```

func vainfo

Returns selected information for the specified virtual address.

```
vainfo (virtaddr selector)
```

Formal Declaration

```
vainfo:any (virtaddr:ptr selector:str)
```

func vtor

Virtual to real. Converts a virtual address to a real address.

```
vtor (virtaddr)
```

Formal Declaration

```
vtor:u32 (virtaddr:ptr)
```

func vtos

Virtual to secondary. Converts a virtual address to a secondary storage address.

```
vtos (virtaddr)
```

Formal Declaration

```
vtos:lptr (virtaddr:ptr)
```

System Debug Environment Variables

The following tables provide short descriptions of all System Debug environment variables, arranged by their logical groups. The information is organized as follows:

Group Name Access Rights Variable Name Return Type

Access rights abbreviations are listed below. PM indicates that privileged mode (PM) capability is required.

r Read access

R	PM read access
w	Write access
W	PM write access
d	Display access (DR command)
D	PM display access (DR command)
m	Modify access (MR command)
M	PM modify access (MR command)

const - constants

```
const   r     FALSE             : BOOL
const   r     TRUE              : BOOL
```

cmd - command related

```
cmd     rw    AUTOIGNORE        : BOOL
cmd     rw    AUTOREPEAT        : BOOL
cmd     rw    CMDLINESUBS       : BOOL
cmd     rw    CMDNUM            : U32
cmd     rw    ECHO_CMDS         : BOOL
cmd     rw    ECHO_SUBS         : BOOL
cmd     rw    ECHO_USE          : BOOL
cmd     rw    ERROR             : S32
cmd     r     MACRO_DEPTH       : U16
cmd     rw    MULTI_LINE_ERRS   : U16
cmd     rw    NONLOCALVARS      : BOOL
cmd     rw    TRACE_FUNC        : U16
```

io - input/output

```
io      rw    CM_INBASE         : STR
io      rw    CM_OUTBASE        : STR
io      r     COLUMN            : U16
io      rW    CONSOLE_IO        : BOOL     (Debug only)
io      rw    FILL              : STR
io      rw    FILTER            : STR
io      rw    HEXUPSHIFT        : BOOL
io      rw    INBASE            : STR
io      rw    JUSTIFY           : STR
io      rw    LIST_INPUT        : BOOL
io      rw    LIST_PAGELEN      : U16
io      r     LIST_PAGENUM      : U16
io      rw    LIST_PAGING       : BOOL
io      rw    LIST_TITLE        : STR
io      rw    LIST_WIDTH        : U16
io      rw    NM_INBASE         : STR
io      rw    NM_OUTBASE        : STR
io      rw    OUTBASE           : STR
io      rw    PROMPT            : STR
io      rw    TERM_KEEPLOCK     : BOOL    (Debug only)
io      rW    TERM_LDEV         : U16     (Debug only)
io      rw    TERM_LOCKING      : BOOL    (Debug only)
io      rw    TERM_LOUD         : BOOL
io      rw    TERM_PAGING       : BOOL
io      rw    TERM_WIDTH        : U16
```

misc - miscellaneous

```
misc    rW    CCODE             : STR     (Debug only)
misc    rW    CSTBASE           : LPTR
misc    r d   CPU               : U16
misc    r     DATE              : STR
misc    r     DISP              : BOOL
misc    rW    DSTBASE           : LPTR
misc    r     ENTRY_MODE        : STR
```

```
misc    rW    ESCAPECODE        : U32     (Debug only)
misc    r     EXEC_MODE         : STR
misc    r     ICSNEST           : U16
misc    r     ICSVA             : LPTR
misc    r     LASTPIN           : U16
misc    rw    LOOKUP_ID         : STR
misc    r     MODE              : STR
misc    r d   MONARCHCPU        : u16
misc    rw    MPEXL_TABLE-VA    : LPTR
misc    r     PIN               : U16
misc    rW    PRIV_USER         : BOOL
misc    r     PROGNAME          : STR
misc    rw    PSTMT             : U16
misc    rw    QUIET_MODIFY      : BOOL
misc    r     SYSVERSION        : STR
misc    r     TIME              : STR
misc    r     VERSION           : STR
```

win - window

```
win     rw    CHANGES           : STR
win     rw    CMPW              : LCPTR
win     r     LW                : LPTR
win     rw    MARKERS           : STR
win     r     NMPW              : LCPTR
win     r     PW                : LCPTR
win     r     PWO               : SPTR
win     r     PWS               : U32
win     r     SHOW_CCTL         : BOOL
win     r     VW                : LPTR
win     r     VWO               : SPTR
win     r     VWS               : U32
win     rw    WIN_LENGTH        : U32
win     rw    WIN_WIDTH         : U32
win     r     ZW                : U32
```

limits - limits for macros and variables

```
limits  rw    MACROS            : U16
limits  r     MACROS_LIMIT      : U16
limits  rw    VARS              : U16
limits  r     VARS_LIMIT        : U16
limits  rw    VARS_LOC          : U16
limits  r     VARS_TABLE        : U16
```

cmreg - compatibility mode regs

```
cmreg   r dm  CIR               : S16
cmreg   r dm  CMPC              : LCPTR
cmreg   r dm  DB                : S16
cmreg   r dm  DBDST             : S16
cmreg   r dm  DL                : S16
cmreg   r d   MAPDST            : S16
cmreg   r d   MAPFLAG           : S16
cmreg   r dm  Q                 : S16
cmreg   r dm  S                 : S16
cmreg   r dm  SDST              : S16
cmreg   r dm  STATUS            : S16
cmreg   r dm  X                 : S16
```

nmreg - native mode regs

```
nmreg   r dm  ARG0 - ARG3       : U32
nmreg   r dM  CCR               : U16
nmreg   r dm  CR0               : U32
nmreg   r dm  CR8 - CR31        : U32
nmreg   r dm  DP                : U32
nmreg   r dM  EIEM              : U32
nmreg   r dM  EIRR              : U32
nmreg   r dM  IIR               : U32
nmreg   r dM  IOR               : U32
nmreg   r dM  IPSW              : U32
nmreg   r dM  ISR               : U32
nmreg   r dM  ITMR              : U32
```

```
nmreg   r dM  IVA              : U32
nmreg   r dm  PC               : LPTR
nmreg   r dm  PCOB             : U32
nmreg   r dm  PCOF             : U32
nmreg   r dm  PCQB             : LPTR
nmreg   r dm  PCQF             : LPTR
nmreg   r dm  PCSB             : U32
nmreg   r dm  PCSF             : U32
nmreg   r dM  PID1 - PID4      : U16
nmreg   r dM  PRIV             : BOOL
nmreg   r d   PSP              : U32
nmreg   r dM  PSW              : U32
nmreg   r d   R0               : U32
nmreg   r dm  R1 - R31         : U32
nmreg   r dM  RCTR             : U32
nmreg   r dm  RET0             : U32
nmreg   r dm  RET1             : U32
nmreg   r d   RP               : U32
nmreg   r dm  SAR              : U16
nmreg   r dm  SL               : U32
nmreg   r dm  SP               : U32
nmreg   r dm  SR0 - SR7        : U32
nmreg   r dM  TR0 - TR7        : U32
```

fpreg - floating point regs

```
 fpreg   r dM  FPSTATUS         : U32
fpreg   r dM  FP0 - FP15       : LPTR    (until S64 is supported)
fpreg   r dM  FPE0 - FPE7      : U32
```

system - system wide debug

```
   system  rW    CONSOLE_DEBUG     : BOOL    (Debug only)
  system  rW    JOB_DEBUG         : BOOL    (Debug only)
  system  rW    DYING_DEBUG       : BOOL    (Debug only)
```

state - process state

The *state* variables consist of all NMREG, CMREG, and FPREG variables.

7 File System

System Defined Files

MPE/iX reserves certain file designators for system defined files. System defined files are reserved words that refer to a specific type of system file.

$STDIN	refers to the device that you used to initiate your current session or job. The device is normally a terminal for a session and spoolfile for a job. Data entries in this file should not have a colon in column 1. (A colon in column 1 indicates the end-of-data). Use the :EOD command to delimit data.
$STDINX	is the same as $STDIN, except that a colon in the first column does not indicate the end of data. Thus $STDINX may contain commands as well as data. Interactive programs and subsystems often use $STDINX to reference the terminal as an input file. Use :EOD or :EOF to indicate the end of data.
$STDLIST	is the device designated as the session or job output device, the device MPE uses to respond to your commands. This device is normally a terminal for sessions and line printer for jobs.
$NULL	is a file designator that is used to tell MPE to read from or write to a non-existent file as though the input-output operation were successful. This file is usually used to discard output.
$NEWPASS	is a temporary disk file. MPE uses $NEWPASS to store information during the execution of a program. When a program closes $NEWPASS, the system automatically changes its name to $OLDPASS.
$OLDPASS	is a temporary disk file containing the contents of the last $NEWPASS file closed. When a $NEWPASS file is renamed $OLDPASS, the system deletes the previous version of $OLDPASS.

You can use $NEWPASS and $OLDPASS when compiling and preparing a program. MPE compilers write object code to $NEWPASS during compilation. When compilation is complete, the compiler closes $NEWPASS and the system renames the object code (USL) file $OLDPASS. When you prepare the USL file ($OLDPASS), the system stores prepared (executable) code in the $NEWPASS file. When preparation is complete, the system closes $NEWPASS and renames the executable code file $OLDPASS. Use the SAVE command to save the program stored in $OLDPASS to a permanent file.

Table 7-1 FFILEINFO File codes

Integer	Mnemonic	Description
0		Default (unreserved)
1024	USL	User subprogram library
1025	BASD	Basic data
1026	BASP	Basic program
1027	BASFP	Basic fast program
1028	RL	Compatibility mode relocatable library
1029	PROG	Compatibility mode program file
1030	NMPRG	Native mode program file
1031	SL	Segmented library
1032	NMSL	Native mode executable library
1033	NMRL	Native mode relocatable library
1035	VFORM	VPLUS forms file
1036	VFAST	VPLUS fast forms file
1037	VREF	VPLUS reformat file
1040	XLSAV	Cross loader ASCII file (SAVE)
1041	XLBIN	Cross loader relocated binary file
1042	XLDSP	Cross loader ASCII file (DISPLAY)
1050	EDITQ	Edit quick file
1051	EDTCQ	Edit KEEPQ file (COBOL)
1052	EDTCT	Edit TEXT file (COBOL)
1054	TDPDT	TDP diary file
1055	TDPQM	TDP proof marked QMARKED
1056	TDPP	TDP proof marked non-COBOL file
1057	TDPCP	TDP proof marked COBOL file
1058	TDPQ	TDP work file
1059	TDPXQ	TDP work file (COBOL)
1060	RJEPN	RJE punch file
1070	QPROC	QUERY procedure file
1080	KSAMK	KSAM key file

Integer	Mnemonic	Description
1083	GRAPH	GRAPH specification file
1084	SD	Self-describing file
1090	LOG	User logging log file
1100	WDOC	HPWORD document
1101	WDICT	HPWORD hyphenation dictionary
1102	WCONF	HPWORD configuration file
1103	W2601	HPWORD attended printer environment
1110	PCELL	IFS 3000/XL character cell file
1111	PFORM	IFS 3000/XL form file
1112	PENV	IFS 3000/XL environment file
1113	PCCMP	IFS 3000/XL compiled character cell file
1114	RASTR	Graphics image in RASTR format
1130	OPTLF	OPT/3000 log file
1131	TEPES	TEPE/3000 script file
1132	TEPEL	TEPE/3000 log file
1133	SAMPL	APS/3000 log file
1139	MPEDL	MPEDCP/DRP log file
1140	TSR	HPToolset root file
1141	TSD	HPToolset data file
1145	DRAW	Drawing file for HPDRAW
1146	FIG	Figure file for HPDRAW
1147	FONT	Reserved
1148	COLOR	Reserved
1149	D48	Reserved
1152	SLATE	Compressed SLATE file
1153	SLATW	Expanded SLATE work file
1156	DSTOR	RAPID/3000 DICTDBU utility store file
1157	TCODE	Code file for Transact/XL compiler
1158	RCODE	Code file for Report/3000 compiler
1159	ICODE	Code file for Inform/3000 compiler

Integer	Mnemonic	Description
1166	MDIST	HPDesk distribution list
1167	MTEXT	HPDesk text
1168	MARPA	ARPA messages file
1169	MARPD	ARPA distribution list
1170	MCMND	HPDesk abbreviated commands file
1171	MFRTM	HPDesk diary free time list
1172	None	Reserved
1173	MEFT	HPDesk external file transfer messages file
1174	MCRPT	HPDesk encrypted item
1175	MSERL	HPDesk serialized (composite) item
1176	VCSF	Reserved
1177	TTYPE	Terminal type file
1178	TVFC	Terminal vertical format control file
1192	NCONF	Network configuration file
1193	NTRAC	Network trace file
1194	NLOG	Network log file
1195	MIDAS	Reserved
1211	ANODE	Reserved
1212	INODE	Reserved
1213	INVRT	Reserved
1214	EXCEP	Reserved
1215	TAXON	Reserved
1216	QUERF	Reserved
1217	DOCDR	Reserved
1226	VC	VC file
1227	DIF	DIF file
1228	LANGD	Language definition file
1229	CHARD	Character set definition file
1230	MGCAT	Formatted application file

Integer	Mnemonic	Description
1236	BMAP	Base map specification file
1242	BDATA	BASIC data file
1243	BFORM	BASIC field order file for VPLUS
1244	BSAVE	BASIC saved program file
1245	BCNFG	Configuration file for default option BASIC program
1258	PFSTA	Pathflow static file
1259	PFDYN	Pathflow dynamic file
1270	RFDCA	Revisable form DCA data stream
1271	FFDCA	Final form DCA data stream
1272	DIU	Document interchange unit file
1273	PDOC	HPWORD/150 document
1401	CWPTX	Reserved
1421	MAP	HPMAP/3000 map specification file
1422	GAL	Reserved
1425	TTX	Reserved
1461	NMOBJ	Native mode object file
1462	PASLB	Pascal/iX source library

Table 7-2 Carriage Control Directives

Octal Code (ASCII)	Description of Carriage Action
%2 - %37 (" ")	Single space (with or without automatic page eject)
%40 - %52 (" ")	Single space (with or without automatic page eject)
%53 ("+")	No space, return (next printing at column 1), cannot be used more than once on the HP 2608A/S without losing data
%54 (" ")	Single space (with or without automatic page eject)
%55 ("-")	Triple space (with or without automatic page eject)
%56 - %57 (" ")	Single space (with or without automatic page eject)
%60 ("0")	Double space (with or without automatic page eject)
%61 ("1 ")	Conditional page eject (form feed) performed by the software; if the printer is not at top-of-form, a page eject is performed. Ignored if: Postspace mode — The current request has a transfer count of 0 and the previous request was `FOPEN`, `HPFOPEN`, `FCLOSE`, or `FWRITE` specifying a carriage-control directive of %61. Prespace mode — Both the current request and the previous request have transfer counts of 0, and the current request and previous request are any combination of `FOPEN`, `HPFOPEN`, `FCLOSE`, or `FWRITE` specifying a carriage-control of %61.
%62	Skip to one line before top of form (valid for HP 2608S and 2563A printers only)
%63	A conditional page eject form feed is performed by the printer; not at top-of-form, a page eject is performed (valid for HP 2608S and 2563A printers only)
%62 - %77 (" ")	Single space (with or without automatic page eject; for terminals)
%104 - %177 (" ")	Single space (with or without automatic page eject; for terminals)
%2*nn*	Space *nn* lines (no automatic page eject); *nn* is any octal number from 0 through 77
%300 - %313	Select VFC Channel 1 - 12 (HP 2613, 2617, 2618, 2619)
%300 - %317	Select VFC Channel 1 - 16 (HP 2608A/S)
%300	Skip to top of form (page eject)
%301	Skip to bottom of form
%302	Single spacing with automatic page eject
%303	Skip to next odd line with automatic page eject
%304	Skip to next third line with automatic page eject

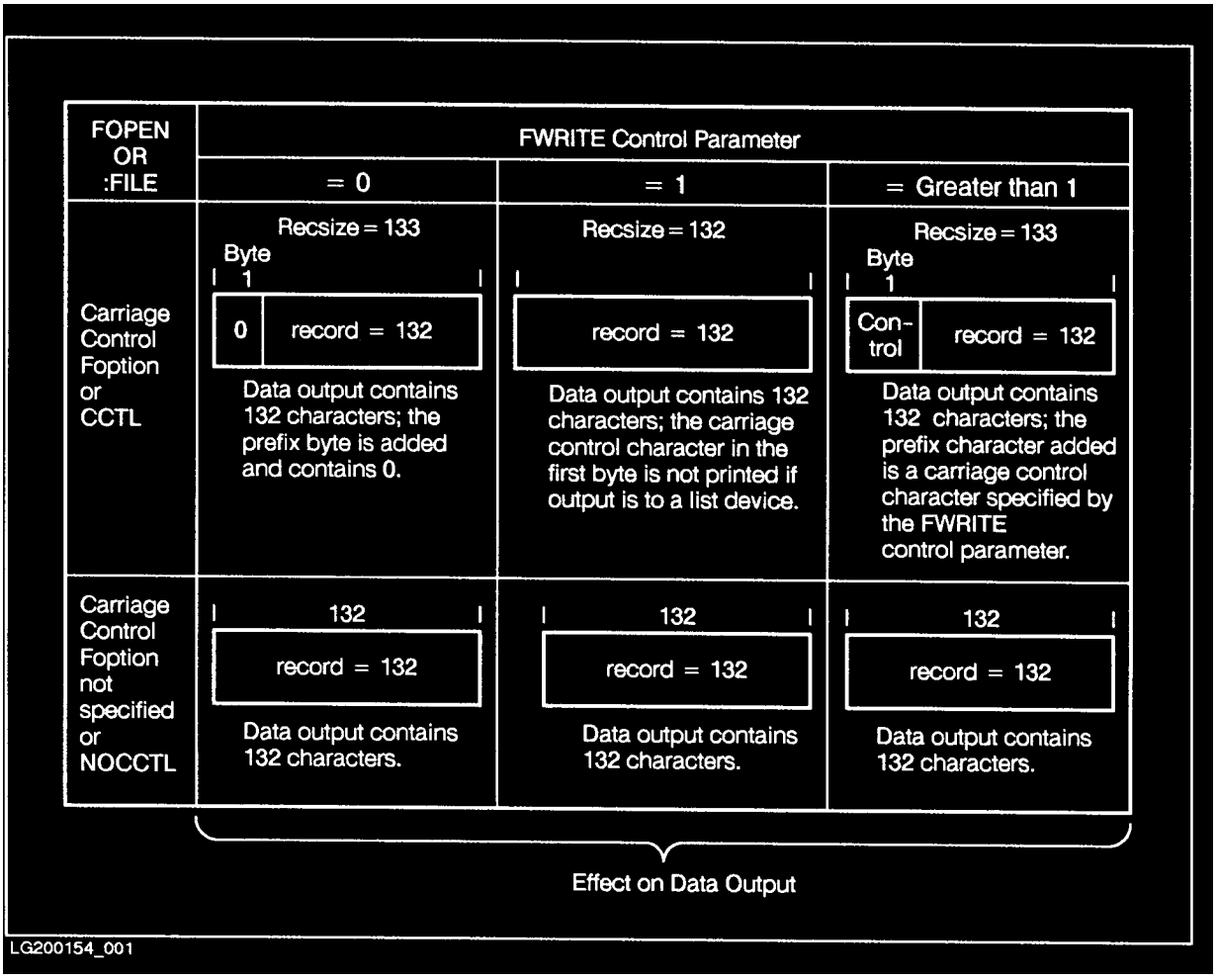
Octal Code (ASCII)	Description of Carriage Action
%305	Skip to next 1/2 page
%306	Skip to next 1/4 page
%307	Skip to next 1/6 page
%310	Skip to bottom of form
%311	User option (HP 2613/17/18/19), skip to one line before bottom of form (HP 2608A/S)
%312	User option (HP 2613/17/18/19), skip to one line before top of form (HP 2608A/S)
%313	User option (HP 2613/17/18/19), skip to top of form (HP 2608A)
%314	Skip to next seventh line with automatic page eject
%315	Skip to next sixth line with automatic page eject
%316	Skip to next fifth line with automatic page eject
%317	Skip to next fourth line with automatic page eject
%310 - %317	(HP 2607)
%314 - %317	(HP 2613/17/18/19)
%320	No space, no return (next printing physically follows this)
%321-%377 (" ")	Single space (with or without automatic page eject)
%400 or %100	Sets postspace movement option (prints first, then spaces). If previous option was prespace movement, the driver outputs a line with a skip to VFC Channel 3 (automatic page eject in effect) or a one line advance (equivalent to an octal code of %201 without automatic page eject) to clear the buffer
%401 or %101	Sets prespace movement option (spaces first, then prints)
%402 or %102	Sets single-space option, with automatic page eject (60 lines per page)
%403 or %103	Sets single-space option, without automatic page eject (66 lines per page)

NOTE If octal codes %55 and %60 are selected with automatic page eject in effect (by default or following an octal code of %102 or %402), the resulting skip is to a location absolute to the page. A code of %60 is replaced by %303, and a code of %55 is replaced by %304. Therefore, the resulting skip can be less than two or three lines, respectively.

If automatic page eject is not in effect, a true double or triple space results, but the perforation between pages is not automatically skipped. For the HP 2608S and 2563A, if auto-eject and feature mode are in

effect, a code of %60 is replaced by two codes of %302, and a code of %55 is replaced by three codes of %302. The resulting skip is double or triple space with auto-eject, respectively.

Carriage Control Effect Summary



File Access and Security

Table 7-3 File Access and Security

Mode	User
R = Read	ANY = Any user
L = Lock (allows exclusive access to the file)	AC = Member of this account only
A = Append (implicitly specifies L also)	GU = Member of this group only
W = Write (implicitly specifies A and L also)	AL = Account librarian user only
X = Execute	GL = Group librarian user only
S = Save	CR = Creating user only

Default Security for Accounts, Groups, and Users

Table 7-4 Security for Accounts, Groups and Users

Type	Access Permitted
SYS account	(R,X:ANY;W,A,L:AC)
Accounts other than SYS	(R,X,W,A,L:AC)
PUB groups in any account	(R,X:ANY;A,W,L,S:AL,GU)
Groups other than PUB	(R,X,S,W,A,L:GU)
Users with no security specified	(R,X,W,A,L:ANY)

Net Default Access to Files

Table 7-5 Net Default Access to Files

Filereference	File	Access Permitted	Save Access to Group
Filename.PUB.SYS	Any file in Public (PUB) group of System (SYS) account.	(R,X:ANY;W:AL,GU)	AL,GU
Filename.grp.SYS	Any file in any group in SYS account.	(R,W,X:GU)	GU
Filename.PUB.acct	Any file in PUB group of any account.	(R,X:AC;W:AL,GU)	AL,GU
Filename.grp.acct	Any file in any group in any account.	(R,W,X:GU)	GU

Capability List

Table 7-6 Capability List

Capability	Mnemonic	Capability	Mnemonic
System manager	= SM	Use private volumes	= UV
Account manager	= AM	Create volumes	= CV
Account librarian	= AL	Use communication	
Group librarian	= GL	software	= CS
Diagnostician	= DI	Programmatic sessions	= PS
System supervisor	= OP	User logging	= LG
Network administrator	= NA	Process handling	= PH
Node manager	= NM	Extra data segments	= DS
Permanent files	= SF	Multiple RINs	= MR
Access of nonshareable		Privileged mode	= PM
I/O devices	= ND	Interactive access	= IA
		Batch access	= BA

Default Capabilities

Table 7-7 Default Capabilities

Type	Capabilities
Accounts	AM,AL,GL,SF,ND,IA,BA
Groups	IA,BA
Users	SF,ND,IA,BA

FOPEN FOPTIONS

FOPEN FOPTIONS

Bits	(0:2)	(2:3)		(5:1)	(6:1)	(7:1)	(8:2)	(10:3)	(13:1)	(14:2)
Field	Reserved	File Type		Disallow :FILE	MPE Tape Labels	Carriage Control	Record Format	Default File Designator	ASCII BINARY	Domain
Meaning		00 00 01 10 11	0=STD 1=KSAM 0=RIO 0=CIR 0=MSG	0 = Allow :FILE 1 = No :FILE	0 = Non-Labeled Tape 1 = Labeled Tape	0 = NOCCTL 1 = CCTL	00 = Fixed 01 = Variable 10 = Undefined 11 = Spoolfile	000 = FILENAME 001 = $STDLIST 010 = $NEWPASS 011 = $OLDPASS 100 = $STDIN 101 = $STDINX 110 = $NULL	0 = BINARY 1 = ASCII	00 = New File 01 = Old Permanent File 10 = Old Temporary File 11 = Old Permanent or Temporary File

LG200154_002

NOTE: Double lines indicate octal digit boundaries.

FOPEN AOPTIONS

FOPEN AOPTIONS

Bits	(0:3)	(3:1)	(4:1)	(5:2)	(7:1)	(8:2)	(10:1)	(11:1)	(12:4)	
Field	Reserved	File Copy Access	NOWAIT I/O	Multi-Access	Inhibit Buffering	Exclusive Access	Dynamic Locking	Multi-Record Access	Access Type	
Meaning		0 = Access Native Mode 1 = Access as Standard Sequential	0 = NOWAIT 1 = Non-NOWAIT	00 = Non Multi-Access 01 = Only Intra-Job Multi-Access 10 = Inter-Job Multi-Access Allowed	0 = BUF 1 = NOBUF	00 = Default 01 = Exclusive 10 = Semi-Exclusive Access Read 11 = Share	0 = No FLOCK Allowed 1 = FLOCK Allowed	0 = No Multi-Record 1 = Multi-Record	0	000 = Read Only
									0	001 = Write Only
									0	010 = Write (Save) Only
									0	011 = Append Only
									0	100 = Read/ Write
									0	101 = Update
									0	110 = Execute

LG200154_003

NOTE: Double lines indicate octal digit boundaries.

8 ASCII Character Set

ASCII Character Set

Table 8-1 ASCII Character Set

Hex.	Dec.	Octal Left	Octal Right	Char
00	0	000000	000000	NUL (null)
01	1	000400	000001	SOH (start of heading)
02	2	001000	000002	STX (start of text)
03	3	001400	000003	ETX (end of text)
04	4	002000	000004	EOT (end of transmission)
05	5	002400	000005	ENQ (enquiry)
06	6	003000	000006	ACK (acknowledge)
07	7	003400	000007	BEL (bell)
08	8	004000	000010	BS (backspace)
09	9	004400	000011	HT (horizontal tabulation)
0A	10	005000	000012	LF (line feed)
0B	11	005400	000013	VT (vertical tabulation)
0C	12	006000	000014	FF (form feed)
0D	13	006400	000015	CR (carriage return)
0E	14	007000	000016	SO (shift out)
0F	15	007400	000017	SI (shift in)
10	16	010000	000020	DLE (data link escape)
11	17	010400	000021	DC1 (device control 1, X-ON)
12	18	011000	000022	DC2 (device control 2)
13	19	011400	000023	DC3 (device control 3, X-OFF)
14	20	012000	000024	DC4 (device control 4)
15	21	012400	000025	NAK (negative acknowledge)
16	22	013000	000026	SYN (synchronous idle)
17	23	013400	000027	ETB (end of transmission block)
18	24	014000	000030	CAN (cancel)
19	25	014400	000031	EM (end of medium)
1A	26	015000	000032	SUB (substitute)

Hex.	Dec.	Octal Left	Octal Right	Char
1B	27	015400	000033	ESC (escape)
1C	28	016000	000034	FS (file separator)
1D	29	016400	000035	GS (group separator)
1E	30	017000	000036	RS (record separator)
1F	31	017400	000037	US (unit separator)
20	32	020000	000040	blank
21	33	020400	000041	!
22	34	021000	000042	"
23	35	021400	000043	#
24	36	022000	000044	$
25	37	022400	000045	%
26	38	023000	000046	&
27	39	023400	000047	' (closing single quote)
28	40	024000	000050	(
29	41	024400	000051	)
2A	42	025000	000052	*
2B	43	025400	000053	+
2C	44	026000	000054	, (comma)
2D	45	026400	000055	-
2E	46	027000	000056	. (period)
2F	47	027400	000057	/
30	48	030000	000060	0
31	49	030400	000061	1
32	50	031000	000062	2
33	51	031400	000063	3
34	52	032000	000064	4
35	53	032400	000065	5
36	54	033000	000066	6
37	55	033400	000067	7
38	56	034000	000070	8

Hex.	Dec.	Octal Left	Octal Right	Char
39	57	034400	000071	9
3A	58	035000	000072	: (colon)
3B	59	035400	000073	; (semicolon)
3C	60	036000	000074	<
3D	61	036400	000075	=
3E	62	037000	000076	>
3F	63	037400	000077	?
40	64	040000	000100	@
41	65	040400	000101	A
42	66	041000	000102	B
43	67	041400	000103	C
44	68	042000	000104	D
45	69	042400	000105	E
46	70	043000	000106	F
47	71	043400	000107	G
48	72	044000	000110	H
49	73	044400	000111	I
4A	74	045000	000112	J
4B	75	045400	000113	K
4C	76	046000	000114	L
4D	77	046400	000115	M
4E	78	047000	000116	N
4F	79	047400	000117	O
50	80	050000	000120	P
51	81	050400	000121	Q
52	82	051000	000122	R
53	83	051400	000123	S
54	84	052000	000124	T
55	85	052400	000125	U
56	86	053000	000126	V

Hex.	Dec.	Octal Left	Octal Right	Char
57	87	053400	000127	W
58	88	054000	000130	X
59	89	054400	000131	Y
5A	90	055000	000132	Z
5B	91	055400	000133	[
5C	92	056000	000134	\
5D	93	056400	000135	]
5E	94	057000	000136	^ (caret)
5F	95	057400	000137	_ (underscore)
60	96	060000	000140	` (opening single quote)
61	97	060400	000141	a
62	98	061000	000142	b
63	99	061400	000143	c
64	100	062000	000144	d
65	101	062400	000145	e
66	102	063000	000146	f
67	103	063400	000147	g
68	104	064000	000150	h
69	105	064400	000151	i
6A	106	065000	000152	j
6B	107	065400	000153	k
6C	108	066000	000154	l
6D	109	066400	000155	m
6E	110	067000	000156	n
6F	111	067400	000157	o
70	112	070000	000160	p
71	113	070400	000161	q
72	114	071000	000162	r
73	115	071400	000163	s
74	116	072000	000164	t

Hex.	Dec.	Octal Left	Octal Right	Char
75	117	072400	000165	u
76	118	073000	000166	v
77	119	073400	000167	w
78	120	074000	000170	x
79	121	074400	000171	y
7A	122	075000	000172	z
7B	123	075400	000173	{
7C	124	076000	000174	| (vertical line)
7D	125	076400	000175	}
7E	126	077000	000176	~ (tilde)
7F	127	077400	000177	DEL delete