

900 Series HP 3000 Computer Systems  
**MPE/iX Intrinsic Reference Manual**



HP Part No. 32650-90028  
Printed in U.S.A. 1995

Sixth Edition Update  
E0195

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.

This document contains proprietary information which is protected by copyright. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

**Copyright © 1995 by Hewlett-Packard Company**

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.

**Restricted Rights Legend**

---

## Printing History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time this document was issued. Many product releases do not require changes to the document. Therefore, do not expect a one-to-one correspondence between product releases and document editions.

<b>Edition</b>	<b>Date</b>	<b>Software Version</b>
First Edition	November 1987	A.01.00
First Edition Update 1	July 1988	A.10.00
First Edition Update 2	December 1988	A.20.00
Second Edition	October 1989	A.20.00
Third Edition	April 1990	A.40.00
Fourth Edition	December 1990	B.30.00
Fifth Edition	June 1992	B.40.00
Sixth Edition	April 1994	C.50.00
Sixth Edition Update	January 1995	C.50.00



---

## 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 PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.



---

## Conventions

**UPPERCASE** In a syntax statement, commands and keywords are shown in uppercase characters. The characters must be entered in the order shown; however, you can enter the characters in either uppercase or lowercase. For example:

**COMMAND**

can be entered as any of the following:

**command          Command          COMMAND**

It cannot, however, be entered as:

**comm              com\_mand          comamnd**

*italics* In a syntax statement or an example, a word in italics represents a parameter or argument that you must replace with the actual value. In the following example, you must replace *filename* with the name of the file:

**COMMAND *filename***

***bold italics*** In a syntax statement, a word in bold italics represents a parameter that you must replace with the actual value. In the following example, you must replace ***filename*** with the name of the file:

**COMMAND(***filename***)**

punctuation In a syntax statement, punctuation characters (other than brackets, braces, vertical bars, and ellipses) must be entered exactly as shown. In the following example, the parentheses and colon must be entered:

**(*filename*):(*filename*)**

underlining Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, yes is the user's response to the prompt:

**Do you want to continue? >> yes**

{ } In a syntax statement, braces enclose required elements. When several elements are stacked within braces, you must select one. In the following example, you must select either **ON** or **OFF**:

**COMMAND { ON }  
          { OFF }**

[ ] In a syntax statement, brackets enclose optional elements. In the following example, **OPTION** can be omitted:

**COMMAND *filename* [OPTION]**

When several elements are stacked within brackets, you can select one or none of the elements. In the following example, you can select **OPTION** or *parameter* or neither. The elements cannot be repeated.

**COMMAND *filename* [ OPTION  
                  *parameter* ]**

---

## Conventions (continued)

[ ... ] In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear within the immediately preceding pair of brackets or braces. In the example below, you can select *parameter* zero or more times. Each instance of *parameter* must be preceded by a comma:

[, *parameter*] [...]

In the example below, you only use the comma as a delimiter if *parameter* is repeated; no comma is used before the first occurrence of *parameter*:

[*parameter*] [, ...]





| ... | In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select **A**, **AB**, **BA**, or **B**. The elements cannot be repeated.




$\left\{ \begin{array}{l} \mathbf{A} \\ \mathbf{B} \end{array} \right\} | \dots |$

... In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.

Δ In a syntax statement, the space symbol Δ shows a required blank. In the following example, *parameter* and *parameter* must be separated with a blank:

(*parameter*)Δ(*parameter*)

 The symbol  indicates a key on the keyboard. For example,  represents the carriage return key or  represents the shift key.

 *character*  *character* indicates a control character. For example, Y means that you press the control key and the Y key simultaneously.



# Contents

---

<b>1. Introduction</b>	
What Is an Intrinsic? . . . . .	1-2
What Is an Intrinsic Mechanism? . . . . .	1-3
<b>2. Intrinsic Use</b>	
Parameters . . . . .	2-1
Parameter Declarations . . . . .	2-1
Parameter Alignment . . . . .	2-2
Parameter Bit Settings . . . . .	2-2
Syntax Statement Mnemonics . . . . .	2-3
Itemnum and Item Pair Options . . . . .	2-4
Data Type Mappings . . . . .	2-5
Functional Returns . . . . .	2-7
Error Handling . . . . .	2-8
Condition Code Errors . . . . .	2-8
Status Parameter Errors . . . . .	2-9
Abort Errors . . . . .	2-12
Additional Capabilities Required . . . . .	2-13
DS Capability . . . . .	2-14
LG Capability . . . . .	2-14
MR Capability . . . . .	2-14
ND Capability . . . . .	2-14
OP Capability . . . . .	2-14
PH Capability . . . . .	2-15
PM Capability . . . . .	2-15
PS Capability . . . . .	2-15
<b>3. Intrinsic Tasks</b>	
<b>4. Command Definitions</b>	
ABORTSESS . . . . .	4-2
ACTIVATE . . . . .	4-4
ADJUSTUSLF . . . . .	4-6
ALMANAC . . . . .	4-8
ALTDSEG . . . . .	4-10
ARITRAP . . . . .	4-12
ASCII . . . . .	4-15
BEGINLOG . . . . .	4-17
BINARY . . . . .	4-19
CALENDAR . . . . .	4-21
CATCLOSE . . . . .	4-22
CATOPEN . . . . .	4-23

CATREAD . . . . .	4-25
CAUSEBREAK . . . . .	4-28
CLEANUSL . . . . .	4-29
CLOCK . . . . .	4-31
CLOSELOG . . . . .	4-32
COMMAND . . . . .	4-34
CREATE . . . . .	4-36
CREATEPROCESS . . . . .	4-43
CTRANSLATE . . . . .	4-48
DASCH . . . . .	4-50
DATELINE . . . . .	4-52
DBINARY . . . . .	4-53
DEBUG . . . . .	4-55
DLSIZE . . . . .	4-56
DMOVIN . . . . .	4-58
DMOVOUT . . . . .	4-61
ENDLOG . . . . .	4-64
EXPANDUSLF . . . . .	4-66
FATHER . . . . .	4-68
FCHECK . . . . .	4-69
FCLOSE . . . . .	4-79
FCONTROL . . . . .	4-84
FDELETE . . . . .	4-107
FDEVICECONTROL . . . . .	4-108
FERRMSG . . . . .	4-130
FFILEINFO . . . . .	4-131
FFINDBYKEY . . . . .	4-151
FFINDN . . . . .	4-153
FGETINFO . . . . .	4-155
FGETKEYINFO . . . . .	4-164
FINDJCW . . . . .	4-166
FINTEXIT . . . . .	4-168
FINTSTATE . . . . .	4-169
FLABELINFO . . . . .	4-171
FLOCK . . . . .	4-182
FLUSHLOG . . . . .	4-184
FMTCALENDAR . . . . .	4-185
FMTCLOCK . . . . .	4-186
FMTDATE . . . . .	4-187
FOPEN . . . . .	4-189
FPARSE . . . . .	4-218
FPOINT . . . . .	4-221
FREAD . . . . .	4-223
FREADBACKWARD . . . . .	4-226
FREADBYKEY . . . . .	4-228
FREADC . . . . .	4-230
FREADDIR . . . . .	4-232
FREADLABEL . . . . .	4-234
FREADSEEK . . . . .	4-236
FREEDSEG . . . . .	4-238
FREELOCRIN . . . . .	4-240

FRELATE . . . . .	4-241
FREMOVE . . . . .	4-243
FRENAME . . . . .	4-244
FSETMODE . . . . .	4-247
FSPACE . . . . .	4-250
FUNLOCK . . . . .	4-252
FUPDATE . . . . .	4-253
FWRITE . . . . .	4-255
FWRITEDIR . . . . .	4-263
FWRITELABEL . . . . .	4-265
GENMESSAGE . . . . .	4-267
GETDSEG . . . . .	4-272
GETINFO . . . . .	4-275
GETJCW . . . . .	4-277
GETLOCRIN . . . . .	4-278
GETORIGIN . . . . .	4-279
GETPRIORITY . . . . .	4-280
GETPRIVMODE . . . . .	4-282
GETPROCID . . . . .	4-283
GETPROCINFO . . . . .	4-284
GETUSERMODE . . . . .	4-286
HP32208 . . . . .	4-287
HPACDINFO . . . . .	4-288
HPACDPUT . . . . .	4-294
HPCICOMMAND . . . . .	4-298
HPCIDELETEVAR . . . . .	4-300
HPCIGETVAR . . . . .	4-301
HPCIPUTVAR . . . . .	4-305
HPDEBUG . . . . .	4-308
HPDEVCONTROL . . . . .	4-310
HPDEVCREATE . . . . .	4-312
HPENBLTRAP . . . . .	4-316
HPERRDEPTH . . . . .	4-319
HPERRMSG . . . . .	4-321
HPERRREAD . . . . .	4-325
HPFDUPLICATE . . . . .	4-327
HPFIRSTLIBRARY . . . . .	4-329
HPFOPEN . . . . .	4-331
HPFPCONVERT . . . . .	4-364
HPGETPROCPLABEL . . . . .	4-368
HPLOADCMPROCEDURE . . . . .	4-371
HPLOADNMPROC . . . . .	4-373
HPMERGEEND . . . . .	4-375
HPMERGEERRORMESS . . . . .	4-376
HPMERGEINIT . . . . .	4-378
HPMERGEOUTPUT . . . . .	4-383
HPMERGESTAT . . . . .	4-384
HPMERGETITLE . . . . .	4-386
HPMYFILE . . . . .	4-387
HPMYPROGRAM . . . . .	4-389
HPPIPE . . . . .	4-392

HPRESETDUMP . . . . .	4-394
HPSELECT . . . . .	4-395
HPSETCCODE . . . . .	4-399
HPSETDUMP . . . . .	4-401
HPSORTEND . . . . .	4-403
HPSORTERRORMESS . . . . .	4-405
HPSORTINIT . . . . .	4-406
HPSORTINPUT . . . . .	4-411
HPSORTOUTPUT . . . . .	4-412
HPSORTSTAT . . . . .	4-414
HPSORTTITLE . . . . .	4-416
HPSWITCHTOCM . . . . .	4-417
HPSWTONMNAME . . . . .	4-419
HPSWTONMPLABEL . . . . .	4-421
HPUNLOADCMPROCEDURE . . . . .	4-423
HPVOLINFO . . . . .	4-425
INITUSLF . . . . .	4-446
IODONTWAIT . . . . .	4-448
IOWAIT . . . . .	4-450
JOBINFO . . . . .	4-452
KILL . . . . .	4-459
LOADPROC . . . . .	4-460
LOCKGLORIN . . . . .	4-462
LOCKLOCRIN . . . . .	4-464
LOCRINOWNER . . . . .	4-466
LOGINFO . . . . .	4-468
LOGSTATUS . . . . .	4-470
MAIL . . . . .	4-472
MERGEEND . . . . .	4-474
MERGEERRORMESS . . . . .	4-475
MERGEINIT . . . . .	4-476
MERGEOUTPUT . . . . .	4-481
MERGESTAT . . . . .	4-482
MERGETITLE . . . . .	4-483
MYCOMMAND . . . . .	4-484
NLAPPEND . . . . .	4-488
NLCOLLATE . . . . .	4-489
NLCOLLATE2 . . . . .	4-491
NLCONVCLOCK . . . . .	4-493
NLCONVCUSTDATE . . . . .	4-495
NLCONVNUM . . . . .	4-497
NLFINDSTR . . . . .	4-500
NLFMTCALENDAR . . . . .	4-502
NLFMTCLOCK . . . . .	4-504
NLFMTCUSTDATE . . . . .	4-506
NLFMTDATE . . . . .	4-508
NLFMTLONGCAL . . . . .	4-510
NLFMTNUM . . . . .	4-512
NLGETLANG . . . . .	4-516
NLINFO . . . . .	4-518
NLJUDGE . . . . .	4-526

NLKEYCOMPARE . . . . .	4-528
NLMATCH . . . . .	4-531
NLMATCHINIT . . . . .	4-532
NLNUMSPEC . . . . .	4-535
NLREPCHAR . . . . .	4-537
NLSCANMOVE . . . . .	4-539
NLSUBSTR . . . . .	4-542
NLSWITCHBUF . . . . .	4-545
NLTRANSLATE . . . . .	4-548
OPENLOG . . . . .	4-550
PAUSE . . . . .	4-552
PRINT . . . . .	4-553
PRINTFILEINFO . . . . .	4-554
PRINTOP . . . . .	4-555
PRINTOPREPLY . . . . .	4-557
PROCINFO . . . . .	4-559
PROCTIME . . . . .	4-565
PUTJCW . . . . .	4-566
QUIT . . . . .	4-568
QUITPROG . . . . .	4-569
READ . . . . .	4-570
READX . . . . .	4-572
RECEIVEMAIL . . . . .	4-574
RESETCONTROL . . . . .	4-576
RESETDUMP . . . . .	4-577
SEARCH . . . . .	4-578
SENDMAIL . . . . .	4-580
SETDUMP . . . . .	4-582
SETJCW . . . . .	4-584
SORTEND . . . . .	4-585
SORTERRORMESS . . . . .	4-586
SORTINIT . . . . .	4-587
SORTINPUT . . . . .	4-592
SORTOUTPUT . . . . .	4-593
SORTSTAT . . . . .	4-595
SORTTITLE . . . . .	4-596
STACKDUMP . . . . .	4-597
STARTSESS . . . . .	4-599
SUSPEND . . . . .	4-604
SWITCHDB . . . . .	4-606
TERMINATE . . . . .	4-607
TIMER . . . . .	4-608
UNLOADPROC . . . . .	4-609
UNLOCKGLORIN . . . . .	4-610
UNLOCKLOCIN . . . . .	4-611
WHO . . . . .	4-612
WRITELOG . . . . .	4-615
XARITRAP . . . . .	4-617
XCONTRAP . . . . .	4-622
XLIBTRAP . . . . .	4-624
XSYSTRAP . . . . .	4-626

ZSIZE . . . . .	4-628
-----------------	-------

**Index**

# Figures

---

- 4-1. DMOVIN Data Movement. . . . . 4-59
- 4-2. DMOVOUT Data Movement. . . . . 4-62
- 4-3. Foption Bit Summary . . . . . 4-149
- 4-4. Aoption Bit Summary . . . . . 4-150
- 4-5. Foption Bit Summary . . . . . 4-162
- 4-6. Aoption Bit Summary . . . . . 4-163
- 4-7. Foption Bit Summary . . . . . 4-181
- 4-8. Carriage Control Byte . . . . . 4-225
- 4-9. Carriage-Control Data . . . . . 4-260

## Tables

---

2-1. Mnemonic Descriptions . . . . .	2-4
2-2. Data Type Mappings . . . . .	2-6
2-3. Condition Code Descriptions . . . . .	2-8
2-4. Status.subsys Parameter Identifiers . . . . .	2-10
2-5. Status.info Parameter Identifiers . . . . .	2-11
2-6. Additional Capabilities Required . . . . .	2-13
3-1. Accessing Files . . . . .	3-2
3-2. Accessing Command Interpreter Features . . . . .	3-3
3-3. Getting System Information . . . . .	3-4
3-4. Managing Processes . . . . .	3-5
3-5. Managing Resources . . . . .	3-5
3-6. Programming for Localization . . . . .	3-6
3-7. Managing Message Catalogs . . . . .	3-6
3-8. Converting Data Types . . . . .	3-7
3-9. Sorting and Merging Data . . . . .	3-7
3-10. Handling Traps . . . . .	3-8
3-11. Managing Logging Facilities . . . . .	3-8
3-12. Debugging Applications . . . . .	3-8
3-13. Obtaining Device Information . . . . .	3-9
3-14. Programming in Privileged mode . . . . .	3-9
3-15. Managing USL Files . . . . .	3-9
3-16. Managing Data Segments . . . . .	3-9
3-17. Changing Stack Size . . . . .	3-9
3-18. Using the Compiler Library . . . . .	3-10
3-19. Programming Switch . . . . .	3-10
3-20. Controlling Asynchronous Devices . . . . .	3-11
4-1. CREATEPROCESS Itemnum/Item Values . . . . .	4-45
4-2. File System Error Codes . . . . .	4-70
4-3. FCONTROL Itemnum/Item Values . . . . .	4-85
4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values . . . . .	4-110
4-5. FDEVICECONTROL Controlcode 192 Values . . . . .	4-128
4-6. FFILEINFO Itemnum/Item Values . . . . .	4-132
4-7. FFILEINFO File Codes . . . . .	4-146
4-8. FLABELINFO Itemnum/Item Values . . . . .	4-174
4-9. FOPEN/HPFOPEN Parameter Equivalents . . . . .	4-216
4-10. Carriage-Control Directives . . . . .	4-258
4-11. HPACDINFO Itemnum1/Item1 Values . . . . .	4-290
4-12. HPACDINFO Itemnum2/Item2 Values . . . . .	4-291
4-13. HPACDPUT Itemnum1/Item1 Values . . . . .	4-295
4-14. HPACDPUT Itemnum2/Item2 Values . . . . .	4-296
4-15. HPCIGETVAR Itemnum/Item Values . . . . .	4-303
4-16. HPCIPUTVAR Itemnum/Item Values . . . . .	4-306
4-17. HPDEBUG Itemnum/Item Values . . . . .	4-309



4-18. HPDEVCONTROL Itemnum/Item Values . . . . .	4-311
4-19. HPDEVCREATE Keywords/Keyvalues . . . . .	4-314
4-20. HPFOPEN Itemnum/Item Values . . . . .	4-333
4-21. . . . .	4-393
4-22. JOBINFO Itemnum/Item Values . . . . .	4-454
4-23. LOGINFO Itemnum/Item Values . . . . .	4-469
4-24. NLINFO Itemnum/Item Values . . . . .	4-520
4-25. PROCINFO Error1,Error2 Values . . . . .	4-560
4-26. PROCINFO Itemnum/Item Values . . . . .	4-561
4-27. STARTSESS Status Returns . . . . .	4-600



## Introduction

---

Many programs use procedures or subroutines to handle recurring tasks. In the MPE/iX operating system many of these tasks are performed by a set of system procedures called intrinsics. Intrinsics are available to any process which has the capabilities required to call the intrinsic.

The tasks that intrinsics handle include:

- Accessing and altering files
- Requesting various utility functions
- Accessing system resources
- Manipulating and managing processes and system resources
- Returning values to a caller through the use of parameters or functional returns

Intrinsics are no different from procedures that you would write for yourself, except that the details of performing a task are invisible.

---

## What Is an Intrinsic?

The term intrinsic refers to any external system or subsystem. However, under MPE/iX this term has a more specific meaning. To qualify as a true Hewlett-Packard documented and user-callable intrinsic, it must meet the following criteria:

- An intrinsic is a Hewlett-Packard supported external interface to an operating system or subsystem service.
- An intrinsic performs type and bounds checks on parameter values before it uses them, thus protecting the operating system and the user from one another.
- An intrinsic is documented in a Hewlett-Packard manual.
- If an intrinsic is enhanced, its interface, capabilities, and feature set remain backward compatible.
- A process may call an intrinsic from any Hewlett-Packard supported programming language.
- An intrinsic differs from other system library procedures

Hewlett-Packard subsystems and applications can also provide interfaces that meet the definition of an intrinsic. Refer to the *MPE/iX Documentation Guide* (32650-90144) for further information.

---

**Note**            You can define routines for access as if they were intrinsics and place them in new or existing intrinsic files and libraries.

---

---

## What Is an Intrinsic Mechanism?

The MPE/iX intrinsics mechanism provides flexible and convenient access to intrinsic routines from various languages. When an external routine is designated as an intrinsic, the compiler uses the intrinsics mechanism to determine how to invoke the routine correctly. For example, a compiler receives information about the number of parameters, the type of each parameter, and the functional return type (if applicable) and does the following:

- Provides a consistent intrinsic interface
- Reduces the burden and error potential of coding multiple external procedure declarations with all the parameters and type declarations required in some languages
- Ensures proper data type conversion
- Verifies proper data alignment
- Provides proper indirect address references to data
- Generates proper reference parameter addresses
- Assigns values to default parameters and correctly resolves optional parameters omitted at the end of an intrinsic call
- Provides parameter checking for languages that have no mechanism for detailed external declarations
- Permits inter-language calls that may not be possible within a particular programming language
- Provides name translation for case-insensitive languages
- Allows future extension without requiring source code changes

For example, Pascal/XL requires a detailed routine header for all external routines, and all calls to a routine must adhere to strict typing constraints. However, by designating a routine as an intrinsic, the compiler uses the intrinsics mechanism to determine how to call the routine which permits removal of the detailed routine header and relaxes the strict typing constraints.

Since the type information, stored in the intrinsic mechanism, is in language-independent form, you can pass parameters of any type that conform functionally to the types expected by the intrinsic. This allows convenient and flexible access to system intrinsics through the intrinsics mechanism.

---

**Note**            You can define routines for access as if they were intrinsics and place them in new or existing intrinsic files and libraries.

---



## Intrinsic Use

---

You invoke an intrinsic by calling it explicitly from within a program, subprogram, or procedure. All system intrinsics are treated as external procedures by user programs and the intrinsics mechanism facilitates the declaration.

To call an intrinsic, you must know the following information:

- The number of parameters and whether any are optional
- The data type of each parameter and whether parameters are passed by value or reference
- The calling sequence or order of the parameters
- The functional return value, if applicable

---

## Parameters

Intrinsic parameters can be input, output, or input/output (I/O) parameters:

- Input parameters pass values into the intrinsic routine.
- Output parameters return values from the intrinsic routine to the calling routine.
- I/O parameters pass values into and return values from the intrinsic routine.

## Parameter Declarations

Parameters are passed to an intrinsic either by value or by reference, depending on how the parameter is declared. When a parameter is passed by reference, its address in the caller's data area is made available to the called procedure. When a parameter is passed by value, the called procedure receives a private copy of the actual data value. If the called procedure changes the private copy, the corresponding value in the calling routine remains unchanged.

Parameters can be either a literal value or a variable; however, only variables can be passed to reference parameters.

A process must call intrinsic parameters in the exact calling sequence (order) specified in the syntax statement and must separate them by commas. For example, write a call to the `CLOSELOG` intrinsic as follows:

```
CLOSELOG(ACCESS,0,0);
```

You must represent optional intrinsic parameters if they are missing. To indicate a missing parameter within the parameter list, omit the parameter, but retain the preceding commas. For example, write a call to the `FOPEN` intrinsic and omit the second (optional) parameter as follows:

```
FILENUM:=FOPEN(MYFILE,,3);
```

If you omit the first parameter from a list, indicate this by following the left parenthesis with a comma (,). If you omit one or more parameters from the end of a list, indicate this by following the last parameter with the right parenthesis; no commas are required.

---

**Note** When a process calls an intrinsic with no parameters from C/XL, follow the intrinsic name with an empty pair of parentheses (()).

---

## Parameter Alignment

For programs running under MPE V and in compatibility mode (CM), parameters are passed by word address, which automatically enforces parameter alignment.

For programs running in native mode (NM), byte addressing is used for all parameters. As a result, it is possible to pass a 16-bit or 32-bit parameter that is not 16-bit or 32-bit aligned. Unless explicitly stated in the parameter description, native mode programs do not impose alignment requirements.

## Parameter Bit Settings

In some intrinsic calls, a process passes input parameters to the intrinsic as words whose individual bits or fields of bits signify functions or options. In cases where bits within a word are described as reserved for the operating system, set those bits to zero. This ensures the compatibility of your current program with future releases of the operating system.

In some intrinsic calls, an intrinsic passes output parameters to words referenced by a calling program. In cases where bits within a word are described as reserved for the operating system, the system sets them to zero unless otherwise noted.



## Syntax Statement Mnemonics

An intrinsic's syntax statement contains the calling sequence, or order, of the parameters, indicates whether a parameter is required or optional, and specifies the parameter type. Syntax statements have the following format:

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

Required parameters, such as *pin* in the example above, appear in ***boldface italics***. Optional parameters, such as *allow*, are shown in *italics*. The mnemonics that appear over the parameters indicate their type and whether they are passed by reference, which is the default, and is not marked, or by value. (In the example, the mnemonics for both parameters have a V, which indicates that they are passed by value.)

Table 2-1 lists the mnemonics used in intrinsic syntax statements and their meanings.

**Table 2-1. Mnemonic Descriptions**

Mnemonic	Meaning
A	Array
B	Boolean
C	Character
CA	Character Array
I16	16-bit signed integer
I32	32-bit signed integer
I64	64-bit signed integer
I*	32-bit signed integer (NM) or 16-bit signed integer (CM)
LA	Logical array
O-P	Option privileged (privileged mode (PM) capability)
REC	Record
R32	32-bit real number
R64	64-bit real number
UDS	User-defined structure
U16	16-bit unsigned integer
U32	32-bit unsigned integer
U64	64-bit unsigned integer
V	Passed by value
@32	32-bit address
@64	64-bit address
@*	32-bit address (NM) or 16-bit address (CM)
*	Type varies

### Itemnum and Item Pair Options

Some intrinsics allow multiple *itemnum, item* pairs to be passed to define one or more options. The *itemnum* parameter is usually an integer value that the intrinsic uses to define the meaning and the expected data type of the value passed or returned in the corresponding *item* parameter.

An *itemnum* cannot be passed without its corresponding *item* value. The only exception is when a special value is passed in the last *itemnum* to indicate the end of the *itemnum, item* list; in this case, an accompanying *item* is not required.

## 2-4 Intrinsic Use

## Data Type Mappings

Table 2-2 maps generic data types to possible implementations in programming languages supported by the operating system. The mnemonics associated with the generic types appear in parentheses. For more information on calling intrinsics from these languages, refer to the appropriate language reference manual.

---

**Note**            The IEEE floating-point standard is the default real format for the 900 Series Hewlett-Packard 3000 computers.

---

**Table 2-2. Data Type Mappings**

Generic Type	COBOL II/XL	FORTRAN 77/XL	Pascal/XL	C & C/XL
16-bit signed integer (I16)	PIC S9(1) - S9(4) BINARY SYNC	Integer*2	SHORTINT or any 16-bit subrange	SHORTINT
32-bit signed integer (I32)	PIC S9(5) - S9(9) BINARY SYNC	Integer or integer*4	Integer or any 32-bit subrange	32-bit
64-bit signed integer (I64)	PIC S9(10) - S9(18) BINARY SYNC	N/A	N/A	N/A
16-bit unsigned integer (U16)	PIC S9(1) - S9(4) BINARY SYNC	Integer*2	0..65535 or any 16-bit subrange	Unsigned short
32-bit unsigned integer (U32)	PIC S9(5) - S9(9) BINARY SYNC	Integer or integer*4	Any 32-bit subrange	Unsigned integer
64-bit unsigned integer (U64)	PIC S9(10) - S9(18) BINARY SYNC	N/A	N/A	N/A
32-bit real (R32)	N/A	Real or real*4	Real	Float
64-bit real (R64)	N/A	Double precision or real*8	Long real	Double
Boolean (B)	Group item	Logical*2	Boolean	SHORTINT
Character (C)	USAGE DISPLAY or group item	Character	Char	Unsigned character or character
32-bit address (@32)	PIC S9(5) - S9(9) BINARY SYNC	Integer or integer*4	LOCALANYPTR or any normal pointer type	Pointer
64-bit address (@64)	PIC S9(10) - S9(18) BINARY SYNC	N/A	GLOBALANYPTR or any pointer type declared with the \$EXTNADDR compiler directive	Long pointer
Array (A)	USAGE DISPLAY or group item	Equivalent array type	Array (any type)	Array (any type)
Record (REC)	USAGE DISPLAY or group item	Array type	Record (any type)	Struct (any type)
User-defined structure (UDS)				

---

## Functional Returns

Certain intrinsics return a value to the calling program in functional returns. These routines return a specified value type (for example, 16-bit signed integer, character, or 64-bit real) in its name.

Functional returns are declared in an assignment statement, where the value returned is assigned to some variable of the same type as the functional return. The symbol `:=` is the assignment operator that is “assigned the value of” or “receives the value of”. The variable receiving the functional return appears on the left-hand side of the assignment operator, while the call to the intrinsic with a functional return appears on the right-hand side.

For example, an intrinsic call description format for functional returns appears below with the `READ` intrinsic, which returns to a 16-bit signed integer indicating the length of the input read. The data type of each variable is signified by a mnemonic above the variable name:

```
I16          U16A      I16V
length:=READ(message,expected);
```

The `READ` intrinsic is read as a 16-bit signed integer procedure, where *message* is a required 16-bit unsigned integer array passed by reference and *expected* is a required 16-bit integer passed by value.

Other expressions that can contain calls to an intrinsic with a functional return can appear in an output statement, the parameter list of a procedure, or a conditional statement. Refer to the individual programmer’s language and reference manuals for details on functional returns.

---

## Error Handling

Three types of errors can occur when an intrinsic is executed:

- Condition code errors. Such errors are generally recoverable since control returns to the calling program. The operating system maintains condition codes to provide backward compatibility with previous (MPE V/E-based) operating systems.
- Status parameter errors. When you include the *status* parameter in the intrinsic call, it returns status information to the calling routine. For some intrinsics, both the condition code and the special status parameter can return information; other intrinsics use only one of these to return information.
- Abort errors. Such errors occur when a calling program passes illegal parameters to an intrinsic or does not have the required capability for the intrinsic. An abort error terminates the process.

### Condition Code Errors

Some intrinsics have an associated condition code. A condition code is a temporary value providing information about what happened during execution. The condition code is affected by many intrinsics and should be checked immediately upon return from an intrinsic.

A condition code has three possible values (CCG, CCL, or CCE) and has the general meaning as indicated in Table 2-3. Specific meanings are listed under the individual intrinsic called.

**Table 2-3. Condition Code Descriptions**

Value	Condition Code	Description
0	CCG (>)	Condition Code Greater Than. A warning or special condition occurred but may not have affected the execution of the request. (For example, the request was executed, but default values were assumed as intrinsic call parameters.)
1	CCL (<)	Condition Code Less Than. The request was not granted, but the error condition may be recoverable.
2	CCE (=)	Condition Code Equal. Generally indicates that the request was granted.

## Status Parameter Errors

Some intrinsics use the *status* parameter to return information on the status of the intrinsic call. This parameter is a native mode (NM) implementation of the condition code feature with extended capabilities. The *status* parameter is a 32-bit signed integer value passed by reference.

---

**Note** It is good programming practice to specify the *status* parameter and check its value after an intrinsic call. If an error or warning condition is encountered and the *status* parameter was not specified, the intrinsic causes the calling process to abort.

---

The *status* parameter is read in two 16-bit fields:

Bits	Value/Meaning
16:16	<i>status.subsys</i> : identifies the subsystem that initiated the error or warning.
0:16	<i>status.info</i> : identifies the type of error or warning. Refer to each intrinsic for possible values.

Table 2-4 and Table 2-5 represent the possible values returned by the *status* parameter.

**Table 2-4. Status.subsys Parameter Identifiers**

<b>Subsystem ID</b>	<b>Subsystem</b>
0	Successful (no errors)
9	CM Loader
098	System Aborts
100	Switch
101	Memory Manager
102	Process Manager
103	Stack Unwind
104	NM Loader
107	Virtual Space Management
108	Table Management
110	Clock/Timer Manager
111	High-Level I/O
113	Low-Level I/O
114	HP-IO Channel Manager
116	HP-IB Adapter Manager
121	CS80 Disk Manager
122	Port (IPC) Facility
123	Dispatcher
127	CIPER Device Manager
128	Terminal I/O Manager for Logging
130	Page Printer Device Manager
133	Measurement Interface
143	File System
145	Secondary Storage Management
151	Transaction Management
153	File System Label Management
154	CM Support Routines
155	Job and Session



**Table 2-4. Status.subsys Parameter Identifiers (continued)**

Subsystem ID	Subsystem
158	Break Handling
161	Command Interpreter
165	Debug Low-Level Breakpoint Management
169	CM Debugger
185	File System Storage Management
187	Error Management
188	CM Emulator, Translator, OCT
193	Floating-Point Conversion
195	Hewlett-Packard SORT
196	Hewlett-Packard MERGE
198	Eagle A-MUX Device Manager
200	U-Code Trap Handler
201	Data Communication Buffer Manager
206	System Logging
213	I/O Services
221	Intrinsic Aborts
222	Intrinsic Names

**Table 2-5. Status.info Parameter Identifiers**

Value	Description
0	Normal
<0	Indicates an error condition
>0	Indicates a warning condition

## Abort Errors

Software traps are interrupts generated by software events in which the operating system interrupts the normal flow of a process' execution. Some intrinsics generate a software trap when they detect certain error conditions. Normally, if an intrinsic causes a trap, the system trap handler aborts the user program. However, you can write a procedure to handle abort errors and use it in place of the default system trap handler, which will permit recovery from errors in certain cases. For more information on handling traps, refer to *Trap Handling Programmer's Guide* (32650-90026).

When a program is aborted in a batch job, the operating system removes the job from the system unless you precede the command that caused the error with a **CONTINUE** command. If the program is aborted in an interactive session, the system returns control to the parent process.

---

## Additional Capabilities Required

Most system intrinsics do not require additional capabilities to access operating system features. However, if an intrinsic requires the operating system to check for an additional capability at program load and/or run time, that capability must be assigned prior to running the program.

Table 2-6 lists additional capabilities and the assignments required by some intrinsics.

---

**Note** A system manager or account manager must assign the capabilities to the group where the program file resides or to the user running the program. You must assign PH, DS, MR, and PM capabilities to the program file at link time using the *caplist* parameter of the LINK command.

---

**Table 2-6. Additional Capabilities Required**

Capability Required	Assignment To
Data segment management (DS)	The program file and the group in which that file resides
User logging (LG)	The user running the program
Multiple RIN (MR)	The program file and the group in which that file resides
Nonshareable device (ND)	The user running the program
System supervisor (OP)	The user running the program
Process handling (PH)	The program file and the group in which that file resides
Privileged mode (PM)	The program file and the group in which that file resides
Programmatic sessions (PS)	The user running the program

The operating system performs two capability checks prior to executing the program. The checks must succeed before the program can run:

- At load time, the file system checks the user capability list against the account-level, group-level, and file-level security provisions for the program file and all referenced executable library (XL) files.
- At load time, the loader checks the program file's capability list against the capability list for the group in which the program file resides. If the program is located in a temporary file, the loader checks the program file's capability list against the capability list for the user who is running the program. In this case, the capability must be assigned to the user in addition to (or instead of) the group.

A third capability check occurs at run time when the program executes the code of an intrinsic that requires a capability. The intrinsic checks either the program file (for DS, MR, ND, PH, or PM) or the user (for LG, OP, or PS). If the operating system does not find the correct capability, the program aborts.

The operating system optional capabilities, and what they allow you to do, are explained below.

### **DS Capability**

The data segment management (DS) capability allows you to create and access extra data segments from processes during a job or session.

### **LG Capability**

The user logging (LG) capability provides a flexible transaction-logging capability that allows you to record additions and modifications to your database and subsystem files on either tape or disk. If the database is lost, the logging tape or disk file facilitates recovery of the lost transactions.

### **MR Capability**

The multiple resource identification number (MR) capability allows you to simultaneously lock as many global resource identification numbers (RINs) as desired.

### **ND Capability**

The nonshareable device (ND) capability allows you to use devices other than terminals, discs, and spooled devices (except for the standard job/session input and list devices). If the device is nonspooled, the user has absolute control of the device.

### **OP Capability**

The system supervisor (OP) capability allows you to optimize the performance of the system and to record additions and modifications to your databases and subsystem files.

## PH Capability

The process handling (PH) capability allows you to do the following programmatically:

- Create and delete processes
- Activate and suspend processes
- Send mail between processes
- Change process scheduling
- Get existing process information

## PM Capability

The privileged mode (PM) capability allows you to access all operating system routines and data structures. The operating system protects privileged users from executing at the same privilege level as the operating system. This prevents privileged users from accessing certain instructions used by the operating system to control the hardware. There are two ways to acquire privileged mode:

- If the routine should always run in privileged mode, use the execution-privilege option of the appropriate compiler and designate the execution level as 2. (Refer to your programming language reference manual for more details.)
- If the routine requires privileged mode only on a selective basis, it is safer to obtain an execution level of 2 only where necessary. To do this, invoke the `GETPRIVMODE` intrinsic.

---

### Caution

The normal checks and limitations that apply to standard users are bypassed in privileged mode. A privileged mode program can destroy file integrity and the operating system software. Hewlett-Packard will investigate and attempt to resolve problems resulting from the use of privileged mode 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 operating system software.

---

## PS Capability

Programmatic sessions (PS) capability allows programmatic creation of sessions on any terminal on the system.



## Intrinsic Tasks

---

This chapter is a quick reference for locating intrinsics that perform designated tasks. Table 3-1 through Table 3-14 describe the general task, a subtask to the general task, and the intrinsic associated with it. The general task is, in most cases, the title of a programmer's guide and the subtasks correlate to chapters within that programmer's guide. For information about the intrinsic specifically, refer to chapter 4 of this manual. For information about the use of an intrinsic, refer to the specified programmer's guide.

The tasks covered are:

- Accessing Files (Table 3-1).
- Accessing Command Interpreter Features (Table 3-2).
- Getting System Information (Table 3-3).
- Managing Processes (Table 3-4).
- Managing Resources and System Information (Table 3-5).
- Programming for Localization (Table 3-6).
- Managing Message Catalogs (Table 3-7).
- Converting Data Types (Table 3-8).
- Sorting and Merging Data (Table 3-9).
- Handling Traps (Table 3-10).
- Managing Logging Facilities (Table 3-11).
- Debugging Applications (Table 3-12).
- Obtaining Device Information (Table 3-13)
- Programming in Privileged mode (Table 3-14).
- Managing USL Files (Table 3-15).
- Managing Data Segments (Table 3-16).
- Changing Stack Size (Table 3-17).
- Using the Compiler Library (Table 3-18).
- Programming Switch (Table 3-19).
- Controlling Asynchronous Devices (Table 3-20).

**Table 3-1. Accessing Files**

Function	Intrinsic	Manual
Opening a file	FOPEN FPARSE HPFOPEN	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Closing a File	FCLOSE	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Writing data to a file	FDEVICECONTROL FSETMODE FUPDATE FWRITE FWRITEDIR FWRITELABEL PRINT PRINTOP PRINTOPREPLY	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Reading data from a file	FREAD FREADBACKWARD FREADDIR FREADSEEK FSETMODE READ READX	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Controlling record pointer movement	FCONTROL FPOINT FSPACE	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Accessing a mapped file	HPFOPEN	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Sharing a file	FOPEN FLOCK FUNLOCK HPFOPEN	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Maintaining file security	FOPEN HPFOPEN  HPACDINFO HPACDPUT	<i>Accessing Files Programmer's Guide (32650-90017)</i>  <i>No manual</i> <i>No manual</i>
Getting file information	FCHECK FERRMSG FFILEINFO FGETINFO FLABELINFO FRELATE FRENAME	<i>Accessing Files Programmer's Guide (32650-90017)</i>



### Accessing Files (Continued)

Function	Intrinsic	Manual
Error Checking	HPERRDEPTH HPERRREAD HPERRMSG PRINTFILEINFO	<i>Accessing Files Programmer's Guide (32650-90017)</i>
Accessing an RIO file	FOPEN FDELETE FREAD FWRITE HPFOPEN	<i>Accessing Files Programmer's Guide (32650-90017)</i>

**Table 3-2. Accessing Command Interpreter Features**

Function	Intrinsic	Manual
Using commands programmatically	COMMAND HPCICOMMAND	<i>Command Interpreter Access and Variables Programmer's Guide (32650-90011)</i>
Controlling variables	HPCIDELETEVAR HPCIGETVAR HPCIPUTVAR	<i>Command Interpreter Access and Variables Programmer's Guide (32650-90011)</i>
Controlling job control words (JCWs)	FINDJCW GETJCW PUTJCW SETJCW	<i>Command Interpreter Access and Variables Programmer's Guide (32650-90011)</i>
Identifying Parameter Input	MYCOMMAND SEARCH	<i>Command Interpreter Access and Variables Programmer's Guide (32650-90011)</i>

**Table 3-3. Getting System Information**

<b>Function</b>	<b>Intrinsic</b>	<b>Manual</b>
Obtaining Time and Date information	ALMANAC CALENDAR CLOCK DATELINE PROCTIME TIMER	<i>Getting System Information Programmer's Guide (32650-90018)</i>
Obtaining File information	FFILEINFO FGETINFO FLABELINFO JOBINFO WHO	<i>Getting System Information Programmer's Guide (32650-90018)</i>
Formatting Time and Date Information	FMTCALENDAR FMTCLOCK FMTDATE	<i>Getting System Information Programmer's Guide (32650-90018)</i>
Error Checking	HPERRDEPTH HPERRREAD HPERRMSG PRINTFILEINFO	<i>Getting System Information Programmer's Guide (32650-90018)</i>

**Table 3-4. Managing Processes**

<b>Function</b>	<b>Intrinsic</b>	<b>Manual</b>
Activating a Process	ACTIVATE	<i>Interprocess Communication Programmer's Guide (32650-90019)</i>
Deactivating/Suspending a Process	ABORTSESS CAUSEBREAK IODONTWAIT IOWAIT KILL PAUSE PROCINFO PROCTIME QUIT QUITPROG STARTSESS SUSPEND TERMINATE	<i>Interprocess Communication Programmer's Guide (32650-90019)</i>
Creating a Process	CREATE CREATEPROCESS	<i>Interprocess Communication Programmer's Guide (32650-90019)</i>
Obtaining Process Information	FATHER GETINFO GETORIGIN GETPRIORITY GETPROCID GETPROCINFO JOBINFO	<i>Interprocess Communication Programmer's Guide (32650-90019)</i>
Obtaining Mail Information	MAIL RECEIVEMAIL SENDMAIL	<i>Interprocess Communication Programmer's Guide (32650-90019)</i>

**Table 3-5. Managing Resources**

<b>Function</b>	<b>Intrinsic</b>	<b>Manual</b>
Managing Global RINs	LOCKGLORIN UNLOCKGLORIN	<i>Resource Management Programmer's Guide (32650-90024)</i>
Managing Local RINs	FREELOCRIN GETLOCRIN LOCKLOCRIN LOCRINOWNER UNLOCKLOCRIN	<i>Resource Management Programmer's Guide (32650-90024)</i>
	HPFIRSTLIBRARY HPGETPROCLABEL HPMYFILE HPMYPROGRAM	<i>Resource Management Programmer's Guide (32650-90024)</i>

**Table 3-6. Programming for Localization**

Function	Intrinsic	Manual
Retrieving information	ALMANAC NLGETLANG NLINFO	<i>Native Language Programmer's Guide (32650-90022)</i>
Handling characters	NLCOLLATE NLFINDSTR NLJUDGE NLKEYCOMPARE  NLMATCH NLMATCHINIT  NLREPCHAR NLSCANMOVE NLSUBSTR NL SWITCHBUF NLTRANSLATE	<i>Native Language Programmer's Guide (32650-90022)</i>
Formatting time and date	NLCONVCLOCK NLCONVCUSTDATE NLFMTCALENDAR NLFMTCLOCK NLFMTCUSTDATE NLFMTDATE NLFMTLONGCAL	<i>Native Language Programmer's Guide (32650-90022)</i>
Formatting numbers	NLCONVNUM NLFMTNUM NLNUMSPEC	<i>Native Language Programmer's Guide (32650-90022)</i>
Using application message catalogs	CATCLOSE CATOPEN CATREAD NLAPPEND	<i>Native Language Programmer's Guide (32650-90022)</i>

**Table 3-7. Managing Message Catalogs**

Function	Intrinsic	Manual
	CATCLOSE CATOPEN CATREAD GENMESSAGE	<i>Message Catalogs Programmer's Guide (32650-90021)</i>

**Table 3-8. Converting Data Types**

Function	Intrinsic	Manual
Converting binary numbers	ASCII DASCII	<i>Data Types Conversion Programmer's Guide (32650-90015)</i>
Converting numeric ASCII strings	BINARY DBINARY	<i>Data Types Conversion Programmer's Guide (32650-90015)</i>
Translating ASCII/EBCDIC or JISCI/EBCDIK	CTRANSLATE	<i>Data Types Conversion Programmer's Guide (32650-90015)</i>
Converting floating-point formats	HPFP_CONVERT	<i>Data Types Conversion Programmer's Guide (32650-90015)</i>

**Table 3-9. Sorting and Merging Data**

Function	Intrinsic	Manual
Creating core merge routines (NM)	HPMERGEEND HPMERGEERRORMESS HPMERGEINIT HPMERGEOUTPUT	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Creating core merge routines (CM)	MERGEEND MERGEERRORMESS MERGEINIT MERGEOUTPUT	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Getting merge information (NM)	HPMERGESTAT HPMERGETITLE	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Getting merge information (CM)	MERGESTAT MERGETITLE	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Creating core sort routines (NM)	HPSORTEND HPSORTERRORMESS HPSORTINIT HPSORTINPUT HPSORTOUTPUT	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Creating core sort routines (CM)	SORTEND SORTERRORMESS SORTINIT SORTINPUT SORTOUTPUT	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Getting sort information (NM)	HPSORTSTAT HPSORTTITLE	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>
Getting sort information (CM)	SORTSTAT SORTTITLE	<i>SORT-MERGE/XL Programmer's Guide (32650-90080)</i>

**Table 3-10. Handling Traps**

Function	Intrinsic	Manual
	ARITRAP FINTEXT FINSTATE HPENABLTRAP RESETCONTROL XARITRAP XCONTRAP XLIBTRAP XSYSTRAP	<i>Trap Handling Programmer's Guide</i> (32650-90026)

**Table 3-11. Managing Logging Facilities**

Function	Intrinsic	Manual
Marking a logical transaction	BEGINLOG ENDLOG	<i>User Logging Programmer's Guide</i> (32650-90027)
Closing a log file	CLOSELOG	<i>User Logging Programmer's Guide</i> (32650-90027)
Flushing the logging buffer	FLUSHLOG	<i>User Logging Programmer's Guide</i> (32650-90027)
Getting information from the log file	LOGINFO LOGSTATUS	<i>User Logging Programmer's Guide</i> (32650-90027)
Opening a log file	OPENLOG	<i>User Logging Programmer's Guide</i> (32650-90027)
Writing to a log file	WRITELOG	<i>User Logging Programmer's Guide</i> (32650-90027)

**Table 3-12. Debugging Applications**

Function	Intrinsic	Manual
Entering the debug facility	DEBUG HPDEBUG	<i>MPE/iX System Debug Reference Manual</i> (32650-90013)
Disarming a debug call	HPRESETDUMP RESETDUMP	<i>MPE/iX System Debug Reference Manual</i> (32650-90013)
Arming a debug call	HPSETDUMP SETDUMP	<i>MPE/iX System Debug Reference Manual</i> (32650-90013)
Producing a full stack trace	STACKDUMP	<i>MPE/iX System Debug Reference Manual</i> (32650-90013)

**Table 3-13. Obtaining Device Information**

Function	Intrinsic	Manual
Obtaining volume information	HPVOLINFO	
Accessing peripheral functionality	HPDEVCONTROL	

**Table 3-14. Programming in Privileged mode**

Function	Intrinsic	Manual
Starting privileged mode	GETPRIVMODE	<i>Introduction to MPE XL for MPE V Programmers (30367-90005)</i>
Ending privileged mode	GETUSERMODE	<i>Introduction to MPE XL for MPE V Programmers (30367-90005)</i>

**Table 3-15. Managing USL Files**

Function	Intrinsic	Manual
Changing USL files	ADJUSTUSLF EXPANDUSLF	<i>MPE Segmenter Reference Manual (30000-90011)</i>
Creating USL files	CLEANUSL	<i>MPE Segmenter Reference Manual (30000-90011)</i>
Initializing USL files	INITUSLF	<i>MPE Segmenter Reference Manual (30000-90011)</i>

**Table 3-16. Managing Data Segments**

Function	Intrinsic	Manual
	ALTDSEG DMOVIN DMOVOUT FREEDSEG GETDSEG SWITCHDB	<i>Introduction to MPE XL for MPE V Programmers (30367-90005)</i>

**Table 3-17. Changing Stack Size**

Function	Intrinsic	Manual
Changing the stack size	DLSIZE ZSIZE	<i>Introduction to MPE XL for MPE V Programmers (30367-90005)</i>

**Table 3-18. Using the Compiler Library**

Function	Intrinsic	Manual
		<i>Compiler Library/XL Reference Manual (32650-90029)</i>

**Table 3-19. Programming Switch**

Function	Intrinsic	Manual
Loading a CM procedure	HPLOADCMPROCEDURE LOADPROC	<i>Switch Programming Guide (32650-90014)</i>
Unloading a CM procedure	HPUNLOADCMPROCEDURE UNLOADPROC	<i>Switch Programming Guide (32650-90014)</i>
Loading a NM procedure	HPLOADNMPLABEL	<i>Switch Programming Guide (32650-90014)</i>
Switching from CM to NM	HPSWTONMNAME HPSWTONMPLABEL	<i>Switch Programming Guide (32650-90014)</i>
Switching from NM to CM	HPSWITCHTOCM HPSETCCODE	<i>Switch Programming Guide (32650-90014)</i>



**Table 3-20. Controlling Asynchronous Devices**

<b>Function</b>	<b>Intrinsic</b>	<b>Manual</b>
Controlling system breaks	<b>FCONTROL</b> <b>CAUSEBREAK</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Controlling subsystem breaks	<b>FCONTROL</b> <b>XCONTRAP</b> <b>RESETCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Specifying carriage control directives	<b>FCONTROL</b> <b>FWRITE</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Specifying an EOR character	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Enabling/Disabling echo	<b>FCONTROL</b> <b>FSETMODE</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Specifying line deletion echo response	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Setting editing mode	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Setting transmission mode	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Specifying and enabling parity	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Specifying terminal type	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Setting a read timeout	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>
Timing a read	<b>FCONTROL</b>	<i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual (32022-90012)</i>



## Command Definitions

---

This chapter describes MPE/iX intrinsics, arranged alphabetically.

Throughout this chapter, paragraphs beginning with the following acronyms should be interpreted as follows:

- (NM) For use in native mode programming only.
- (CM) For use in compatibility mode programming only.
- (KSAM) For use with KSAM files only.
- (ASC) For use with asynchronous serial communications only.
- (SPL) For use with SPL programming language only.

---

## ABORTSESS

NM and CM callable.

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

### Syntax

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

### Parameters

*jsid*            **16-bit signed integer by value (required)**

Indicates whether the *jsnum* parameter refers to a job number or a session number:

Value	Meaning
1	Session number
2	Job number

Because jobs and sessions can have identical numbers, the *jsid* parameter is required to indicate whether *jsnum* describes a job or a session.

*jsnum*            **32-bit signed integer by value (required)**

Passes the job or session number of the job/session to be aborted.

*jsstatus*        **16-bit signed integer array (required)**

Returns a two-element array containing status information. The second element is reserved for the operating system. The first element returns one of the following status values:

Value	Meaning
0	Successful
1	Jobsecurity set to high, or job/session not yours
2	Job/session does not exist
3	Job/session being introduced and cannot be aborted when in the INTRO state

## Operation Notes

All resources held by the aborted job/session are released, and any I/O activity is terminated. Although the job/session is abnormally terminated, the operating system issues log records and updates CPU and connect times. The following message is displayed on the job/session list device:

```
SESSION ABORTED BY SYSTEM MANAGEMENT
```

**ABORTSESS** can be applied to waiting or scheduled jobs, or executing jobs or sessions. If the spooler input file (**\$STDIN**) for a batch job has been created and has not yet been opened, the entire file is deleted. If **ABORTSESS** is issued before the output spoolfile is complete, only that portion of the file already opened is printed and an error message is issued indicating that the job was aborted.

**ABORTSESS** can be successfully called only if the program is executing in a job/session that satisfies at least one of the following conditions:

- The job/session is on the system console.
- The job/session was allowed the **ABORTJOB** command.
- **JOBSECURITY** is set to **LOW** and the user and account names of the job/session are the same as the user and account names of the job/session being aborted.
- **JOBSECURITY** is set to **LOW** and the job/session has account manager (AM) capability, and the account name of the job/session is the same as the account name of the job/session being aborted.
- **JOBSECURITY** is set to **LOW** and the job/session has system manager (SM) capability.

## Related Information

Intrinsics      **STARTSESS**

Commands      None

Manuals      *Process Management Programmer's Guide (32650-90023)*

---

## ACTIVATE

NM and CM callable.

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

### Syntax

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

### Parameters

*pin*            **16-bit signed integer by value (required)**

Indicates the parent or child process to be activated. To indicate a parent process, set *pin* to zero. To indicate a child process, set *pin* to the process identification number (PIN) of the process. The called process must always be expecting an activation from the caller (refer to the `SUSPEND` and `CREATE` intrinsics).

*allow*            **16-bit unsigned integer by value (optional)**

Passes suspend or execute information to the process being activated:

- When *allow* is not specified or is zero, the called process is activated by the operating system but does not commence execution immediately. Instead, control returns to the calling process, which continues execution. This is the default.
- When *allow* is specified and is not zero, the calling process is suspended while the called process is activated and commences execution. Bits (14:2) specify the anticipated source of the call that reactivates the calling process:

Bits	Value/Meaning
15:1	Parent activation bit: <ul style="list-style-type: none"><li>0 Does not expect activation by a parent</li><li>1 Expects activation by a parent</li></ul>
14:1	Child activation bit: <ul style="list-style-type: none"><li>0 Does not expect activation by a child</li><li>1 Expects activation by one child</li></ul>
0:14	Reserved for the operating system

## Operation Notes

A newly created process can only be activated by its parent. A suspended process can be reactivated by its parent, or any of its children, as specified in the *allow* parameter of the ACTIVATE and SUSPEND intrinsics. When a process is activated, it runs until it is either suspended or deleted.

The ACTIVATE intrinsic aborts the calling process (and possibly the entire job/session) if a request to activate the parent results in the activation of the root CI or a system process. The abort error messages associated with these situations are:

```
ACTIVATION OF SYSTEM PROCESS NOT ALLOWED
(ACTIVATE ERROR 20)
```

```
ACTIVATION OF MAIN PROCESS NOT ALLOWED
(ACTIVATE ERROR 21)
```

## Condition Codes

- CCE (2) Request granted. The called process is activated. The calling process is suspended if you specified *allow*.
- CCG (0) Request granted. The called process is already active. The calling process is suspended if you specified *allow*.
- CCL (1) Request denied. The called process was not expecting activation by this calling process, an illegal *pin* parameter was specified, or the *allow* parameter was specified improperly.

## Related Information

Intrinsics CREATE, SUSPEND

Commands None

Manuals *Process Management Programmer's Guide* (32650-90023)

---

## 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.

---

**Note** A USL contains CM object code and is meaningful only in the CM program development process.

---

### Syntax

I16	I16V	I16V
<i>uslerror</i> := ADJUSTUSLF( <i>uslfnm</i> , <i>adjustment</i> );		

### Functional Return

*uslerror*      **16-bit signed integer (assigned functional return)**

Returns the error number if an error occurs (condition code returns CCL (1)). If no error occurs, no value is returned. The error number returned corresponds to the following:

Value	Meaning
0	The file specified by <i>uslfnm</i> was empty, an unexpected end-of-file was encountered while reading the <i>uslfnm</i> , or an unexpected end-of-file was encountered while writing the <i>uslfnm</i> .
1	Unexpected I/O error occurred.
4	Request exceeds the maximum file directory size (32,768 half words).
5	Insufficient directory space.
6	Insufficient USL file information block space.



**Parameters**

<i>uslfnm</i>	<b>16-bit signed integer by value (required)</b> Passes the file number of the USL file (returned by FOPEN/HPFOPEN).
<i>increment</i>	<b>16-bit signed integer by value (required)</b> Returns the assigned record count: <ul style="list-style-type: none"> <li>■ If a positive value, the information block is moved toward the end-of-file in the USL file, increasing the directory block and decreasing the information block.</li> <li>■ If a negative value, the information block is moved toward the start of the USL file, decreasing the directory block and increasing the information block.</li> </ul>

**Condition Codes**

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. An error number was returned to <i>uslerror</i> indicating the reason for this failure.

**Related Information**

Intrinsics	CLEANUSL, EXPANDUSLF, INITUSLF
Commands	None
Manuals	<i>MPE Segmenter Reference Manual</i> (30000-90011)

---

## 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.

### Syntax

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

### Parameters

*date*            **16-bit unsigned integer by value (required)**

Contains the date in the following format:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*dateerror*      **16-bit unsigned integer array (required)**

Returns two elements; the first element is the error number, the second element is reserved and always returns 0. The possible error numbers returned are:

Value	Meaning
0	Successful
1	No parameters available for returning values
2	Day of year out of range
3	Year of century out of range

*yearnum*        **16-bit signed integer by reference (optional)**

Returns the year of century. For example, 00=1900, 84=1984.

*monthnum*      **16-bit signed integer by reference (optional)**

Returns the month of year. For example, 1=January, 12=December.

*daynum*        **16-bit signed integer by reference (optional)**

Returns the day of month.



---

## 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.

---

**Note** Data segment management (DS) intrinsics are not recommended for use in the operating system native mode programming environment; use of DS intrinsics degrade program performance.

---

### Syntax

U16V    I16V    I16 ALTDSEG( <i>index,increment,size</i> );
--

### Parameters

*index*      **16-bit unsigned integer by value (required)**  
Indicates the index of the extra data segment, obtained from the GETDSEG call.

*increment*      **16-bit signed integer by value (required)**  
Specifies the value, in half words, the extra data segment is to be changed. A positive value requests an increase; a negative value requests a decrease.

*size*      **16-bit signed integer by reference (required)**  
Returns the new size of the extra data segment after incrementing or decrementing has occurred.

## Operation Notes

Incrementation and decrementation is accomplished in even multiples of four half words, which are rounded up. For example (in half words):

Present Segment Size	Change Value	New Segment Size
128	-3	128
128	-4	124
128	+1	132
128	+3	132
128	+4	132

When a data segment is created through `GETDSEG`, the required virtual space is allocated by the system to accommodate the original length of the data segment. This virtual space is allocated in increments of 512 half words. For example, creation of an extra data segment with a length of 600 half words results in the allocation of 1024 half words of virtual space.

---

**Note** ALTDSEG cannot increase the size of a data segment to exceed the virtual space originally allocated through `GETDSEG`.

---

When `GETDSEG` is called in privileged mode, the `ALTDSEG` intrinsic must also be called in privileged mode.

## Condition Codes

- CCE (2) Request granted.
- CCG (0) Request granted. An illegal decrement was requested to a total segment size of  $\leq 0$ ; the current size remains in affect. An illegal increment was requested for a total segment size greater than the virtual space originally assigned; the size of the virtual space is granted and returned to the *size* parameter.
- CCL (1) Request denied. An illegal *index* parameter was specified.

## Additional Discussion

- Intrinsics `FREEDSEG`, `GETDSEG`
- Commands None
- Manuals *Introduction to MPE XL for MPE V Programmers (30367-90005)*

---

## ARITRAP

NM and CM callable (differences noted below).

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

### Syntax

I\*V  
ARITRAP(*trapstate*);

### Parameters

*trapstate* (NM) **32-bit signed integer by value (required)**  
(CM) **16-bit signed integer by value (required)**

(NM) Enables/disables arithmetic traps:

Value	Meaning
0	Disable all arithmetic traps
1	Enable all arithmetic traps (except the IEEE inexact result trap)

(CM) Enables/disables CM arithmetic traps.

### Operation Notes

There is a difference between arming and enabling traps. Enabling a trap means that the occurrence of a trap condition is not ignored. Arming a trap is required so that, on a trap condition, a user-written routine is invoked and can take appropriate recovery actions. The following list summarizes what can happen when an arithmetic trap condition occurs:

- If a trap is both enabled and armed, the user-written trap handler is invoked.
- If a trap is enabled but not armed, one of two situations apply:
  - If a Pascal/XL TRY statement was executed, pass control to the RECOVER block by doing an ESCAPE.
  - If a Pascal/XL TRY statement was not executed, an error message is output and the process aborts.
- If a trap is disabled, even though it is armed, the trap is ignored, and execution of the process continues without any interruption.

---

**Note** By default, all traps (except the IEEE inexact result trap) are enabled and the system trap handler is armed.

---

(NM) The possible interrupts listed below are collectively called the arithmetic traps:

- 3000 mode double-precision divide by zero
- 3000 mode double-precision overflow
- 3000 mode double-precision underflow
- 3000 mode floating-point divide by zero
- 3000 mode floating-point overflow
- 3000 mode floating-point underflow
- 3000 mode packed decimal error
- Decimal divide by zero
- Decimal overflow
- IEEE floating-point divide by zero
- IEEE floating-point underflow
- IEEE floating-point overflow
- IEEE floating-point, invalid operation
- Integer divide by zero
- Integer overflow
- Invalid ASCII digit
- Invalid decimal digit
- Paragraph stack overflow
- Range errors
- Result of software-detected pointer arithmetic misaligned or error in conversion from long pointer to short pointer
- Software-detected NIL pointer reference
- Unimplemented condition traps

---

**Note**            The IEEE inexact result trap is not enabled by ARITRAP.

---

(CM) The possible interrupts listed below are collectively called the arithmetic traps:

- 3000 mode double-precision divide by zero
- 3000 mode double-precision overflow
- 3000 mode double-precision underflow
- 3000 mode floating-point divide by zero
- 3000 mode floating-point overflow
- 3000 mode floating-point underflow
- Decimal divide by zero
- Decimal overflow
- Integer divide by zero
- Integer overflow
- Invalid ASCII digit
- Invalid decimal digit
- Invalid decimal operand length
- Invalid source word count

## ARITRAP

### Condition Codes

CCE (2)	Request granted. All arithmetic traps were originally disabled.
CCG (0)	Request granted. At least one arithmetic trap was originally enabled.
CCL (1)	Not returned.

### Related Information

Intrinsics	XARITRAP
Commands	None
Manuals	<i>Trap Handling Programmer's Guide</i> (32650-90026)



## ASCII

NM and CM callable.

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

### Syntax

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

### Functional Return

*numchar*      **16-bit signed integer (assigned functional return)**

Returns the number of characters in the resulting ASCII equivalent.

### Parameters

*binvalue*      **type varies by value (required)**

Passes the binary number to be converted to an ASCII string:

- For octal conversions, *binvalue* must be a 16-bit unsigned integer.
- For decimal conversions, *binvalue* must be a 16-bit signed integer.
- For hexadecimal conversions, *binvalue* must be a 16-bit unsigned integer

*base*            **16-bit signed integer by value (required)**

Passes and must be one of the following values or the process aborts:

Value	Meaning
8	Convert to octal (pad with zeros)
10	Convert to decimal (left-justify)
-10	Convert to decimal (right-justify)
16	Convert to hexadecimal (pad with zeros)

*asciieqv*        **character array (required)**

Returns the converted value. Must be long enough to contain the result (<= 6 characters):

- For octal conversions (*base*=8), 6 characters (including leading zeros) are returned; *numchar* returns the number of significant (right-justified) characters (excluding leading zeros). If *binvalue*=0, the length returned is 1.

## ASCII

- For decimal conversions, *binvalue* is considered a 16-bit, twos complement integer ranging from -32768 to +32767. If *binvalue*=0, only one zero character is returned in *asciieqv*; *numchar* returns the total number of characters (including the sign). For example, if *binvalue*=0, the length returned is 1; and if *binvalue*=327, the length returned is 3.
- For decimal left-justified conversions (*base*=10), leading zeros are removed, and the numeric ASCII result is left-justified in *asciieqv*; the most significant digit (or the "-" sign) is in *asciieqv*(1), the next most significant digit is in *asciieqv*(2), and so on.
- For decimal right-justified conversions (*base*=-10), the result is right-justified in *asciieqv*; the least significant digit is in *asciieqv*(-1), the next least significant digit is in *asciieqv*(-2), and so on.
- For right-justified conversions, the character array where the converted value is to be placed must specify the rightmost byte where data is placed. For example, if *asciieqv* is a 10 byte array declared as:

```
VAR  
MYSTRING : ARRAY [1..10] OF CHAR;
```

then you must specify it in the ASCII intrinsic call as follows (for right justification):

```
NUMCHAR:=ASCII(VALUE,-10&  
,WADDRESS(MYSTRING[10]));
```

The result is right-justified in *asciieqv*, with the rightmost digit of the result contained in the last (rightmost) byte of *asciieqv*.

- For hexadecimal conversions (*base*=16), 4 characters (including leading zeros) are returned. The digits can be 0..9 and A..F. *Numchar* returns the number of significant (right-justified) characters (excluding leading zeros). For example, if *binvalue*=32, *numchar* is 2 and *asciieqv* will be 0020.

---

**Note** For all right-justified conversions *asciieqv* must be initialized to blanks before the call is made.

---

### Related Information

Intrinsics	DASCII, BINARY, DBINARY
Commands	None
Manuals	<i>Data Types Conversion Programmer's Guide</i> (32650-90015)

## 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.

### Syntax

I32 U16A I16 I16 I16 <b>BEGINLOG(<i>index,data,length,mode,logstatus</i>)</b>
--

### Parameters

- index***      **32-bit signed integer by reference (required)**  
 Passes access information to the user logging system. This is the value returned in the ***index*** parameter of the **OPENLOG** intrinsic.
- data***      **16-bit unsigned integer array (required)**  
 Passes the information to be logged. A log record contains 128 half words where 119 are available for writing data. For the most efficient use of log file space, structure arrays with lengths in multiples of 119 half words.
- length***      **16-bit signed integer by reference (required)**  
 Passes the length of the data. A positive value indicates half words; a negative value indicates bytes. If the length is greater than 119 half words (or 238 bytes), the information in ***data*** is divided into two or more physical log records.
- mode***      **16-bit signed integer by reference (required)**  
 Passes a value indicating whether the logging process should suspend the process if it cannot complete the request for service immediately. If it is not possible to log the transaction and ***mode*** is set to **nowait**, the **BEGINLOG** intrinsic indicates, through ***logstatus***, that it could not complete the request:

Value	Meaning
0	Wait
1	Nowait

## BEGINLOG

*logstatus*      16-bit signed integer by reference (required)

Returns a value, indicating the success/failure of the intrinsic call:

Value	Meaning
0	Successful
1	Nowait mode requested, and user logging process is busy
2	Parameter out of bounds
4	Incorrect <i>index</i> parameter passed
5	Incorrect <i>mode</i> parameter passed
9	Error occurred while writing
12	System out of disk space, user logging cannot proceed
14	Invalid access
15	End-of-file

### Operation Notes

Use the *data* parameter of this intrinsic to post user data to the logging file. This use of **BEGINLOG** is similar to using the **WRITELOG** intrinsic, but also denotes the beginning of a logical transaction.

### Related Information

Intrinsics	ENDLOG, WRITELOG
Commands	None
Manuals	<i>User Logging Programmer's Guide</i> (32650-90027)

## BINARY

NM and CM callable.

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

### Syntax

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

### Functional Return

*bineqv*      **16-bit signed integer (assigned functional return)**

Passes the twos complement binary equivalent of the numeric string.

### Parameters

*asciieqv*      **character array (required)**

Passes the octal or signed-decimal number (in ASCII characters) to be converted. If the character string begins with a percent sign (%), it is treated as an octal value. If the character string begins with a dollar sign (\$), it is treated as a hexadecimal value. In this case, the string must be less than 6 characters and only 0 through 9, a through f, and A through F are allowed. If the character string begins with a plus sign, a minus sign, or a number, it is treated as a decimal value. Leading blanks are not allowed and are treated as illegal characters.

*length*      **16-bit signed integer by value (required)**

Passes the length (number of bytes) of the ASCII-coded value. If the value of *length* is 0, the intrinsic returns 0 to the calling process. When the value of *length* is negative, the intrinsic causes the process to abort.

### Condition Codes

- CCE (2)      Request granted. A 16-bit binary value is returned to the process.
- CCG (0)      Request denied. A word overflow, possibly resulting from too many characters (*asciieqv* number too large), occurred in the value returned.
- CCL (1)      Request denied. An illegal character was encountered in *asciieqv*. For example, the digits 8 or 9 were specified in an octal value.

## **BINARY**

### **Related Information**

Intrinsics      DBINARY

Commands      None

Manuals      *Data Types Conversion Programmer's Guide (32650-90015)*

---

## CALENDAR

NM and CM callable.

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

### Syntax

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

### Functional Return

*date*            **16-bit unsigned integer (assigned functional return)**

Returns the calendar date in the following format:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

### Related Information

Intrinsics        ALMANAC

Commands        None

Manuals         *Getting System Information Programmer's Guide* (32650-90018)

---

## CATCLOSE

NM and CM callable.

Closes an application message catalog that was opened with CATOPEN.

### Syntax

I32V      U16A CATCLOSE( <i>catindex</i> , <i>catstatus</i> )
--

### Parameters

*catindex*      **32-bit signed integer by value (required)**

Passes the catalog index returned by the CATOPEN intrinsic.

*catstatus*      **16-bit unsigned integer array (required)**

Returns two elements; the first element is the error number, the second element is reserved and always returns 0. The possible values of the first element returned are:

Value	Meaning
0	Successful
1	Close of catalog file failed
100	Internal message facility error

### Related Information

Intrinsics      CATOPEN, CATREAD

Commands      None

Manuals      *Message Catalogs Programmer's Guide* (32650-90021)



---

## 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.

### Syntax

I32	CA	U16A
<i>catindex</i> :=CATOPEN( <i>formaldesig</i> , <i>catstatus</i> );		

### Functional Return

*catindex*      **32-bit signed integer (assigned functional return)**

An internal value recognized by the CATREAD and CATCLOSE intrinsics.

---

**Note**            The functional return is not a file number.

---

### Parameters

*formaldesig*    **character array (required)**

Passes a string of ASCII characters that identify the catalog file to the system. This string must be terminated by an ASCII special character, it does not accept “/”, “.”, “:”, or “!” as terminators.

*formaldesig* does not accept “/”, “.”, “:”, or “!” as terminators.

*catstatus*      **16-bit unsigned integer array (required)**

Returns two elements; the first element is the error number, the second element is reserved and always returns 0. The possible values of the first element returned are:

Value	Meaning
0	Successful
1	Open failed on catalog file
2	Could not access catalog file
3	File specified not a GENCAT formatted catalog
100	Internal message facility error

## **CATOPEN**

### **Operation Notes**

The CATOPEN intrinsic may alter the condition code (CCODE).

### **Related Information**

Intrinsics      CATCLOSE, CATREAD

Commands      None

Manuals      *Message Catalogs Programmer's Guide* (32650-90021)

## 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.

### Syntax

```

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

```

### Functional Return

*msglength*     **16-bit signed integer (assigned functional return)**  
Returns the length of the message (in bytes).

### Parameters

*catindex*     **32-bit signed integer by value (required)**  
Passes the index (returned by the CATOPEN intrinsic) that specifies the message catalog to be used. The message catalog must have been formatted with the GENCAT utility.

*setnum*       **16-bit signed integer by value (required)**  
Passes the message set number within the catalog (1..255).

*msgnum*       **16-bit signed integer by value (required)**  
Passes the message number within the message set (0..32,766).

## CATREAD

*catstatus* **16-bit unsigned integer array (required)**

Returns two elements; the first element is the error number, the second element is reserved and always returns 0. The possible values of the first element returned are:

Value	Meaning
0	Successful
1	Invalid <i>catindex</i> specified
2	Read failed on catalog file
3	Message set not found
4	Message number not found
6	User buffer overflow
7	Write to <i>msgdest</i> file failed
14	<i>sctnum</i> $\leq 0$ specified
15	<i>sctnum</i> $> 255$ specified
16	<i>msgnum</i> $< 0$ specified
17	<i>msgnum</i> $> 32,766$ specified
18	<i>bufferize</i> $\leq 0$ specified
19	<i>msgdest</i> $\leq -2$ specified
100	Internal message facility error

*buffer* **character array (optional)**

Returns the assembled message.

*bufferize* **16-bit signed integer by value (optional)**

Passes the buffer length in bytes if you specify *buffer*. Passes the length (in bytes) of the message to be written to the destination file if *buffer* is not specified.

Default = 72 bytes.

*parm1-parm5* **character arrays (optional)**

Passes parameters to be inserted into the message. The parameter must always point to a character string and the character string must be terminated by a binary zero.

*msgdest*      **16-bit signed integer by value (optional)**

Passes the file number of the message's destination file (0 or -1 = \$STDLIST, >2 = file number of the destination file). If *buffer* or *msgdest* are not specified, the message goes to \$STDLIST (\$STDLIST = 0).

### Operation Notes

The combined total size of the *buffer* and *parm1* through *parm5* parameters cannot exceed 64,000 bytes.

### Related Information

Intrinsics      CATCLOSE, CATOPEN

Commands      None

Manuals      *Message Catalogs Programmer's Guide* (32650-90021)

---

## 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, and is not applicable in jobs. The program is suspended while in break mode. Execution of the program can resume where the interruption occurred by entering the **RESUME** command, or be aborted by entering the **ABORT** command.

### Syntax

```
CAUSEBREAK;
```

### Condition Codes

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. The intrinsic was not called from an interactive session.

### Related Information

Intrinsics	None
Commands	<b>ABORT</b> , <b>RESUME</b>
Manuals	<i>Process Management Programmer's Guide</i> (32650-90023)

---

## 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.

---

**Note** A USL contains CM object code and is meaningful in the CM program development process only.

---

### Syntax

I16	I16V	CA
<i>filenum</i> := CLEANUSL( <i>uslfnm</i> , <i>formaldesig</i> );		

### Functional Return

*filenum* **16-bit signed integer (assigned functional return)**

Returns the new file number, if no error occurs, or if an error occurs condition code returns CCL (1) and *filenum* returns one of the following:

Value	Meaning
0	Unexpected end-of-file (EOF) marker on either the old or the new USL file
1	Unexpected I/O error on either the old or the new USL file
7	Unable to open new USL file
12	Invalid USL file

### Parameters

*uslfnm* **16-bit signed integer by value (required)**

Returns the file number of the file.

*formaldesig* **character array (required)**

Passes the name to be given to the cleaned file. The array must end with a blank. If it contains all blanks, it deletes the inactive entries.

## **CLEANUSL**

### **Condition Codes**

- CCE (2) Request granted. The new file number is returned to *filenum*.  
CCG (0) Not returned.  
CCL (1) Request denied. An error number is returned to *filenum*.

### **Related Information**

- Intrinsics ADJUSTUSLF, EXPANDUSLF, INITUSLF  
Commands None  
Manuals *MPE Segmenter Reference Manual (30000-90011)*



## CLOCK

NM and CM callable.

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

### Syntax

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

### Functional Return

*time*            **32-bit signed integer (assigned functional return)**

Returns the time, as monitored by the system timer, in the following format:

Bits	Value/Meaning
24:8	Tenths of seconds
16:8	Seconds
8:8	Minute of hour
0:8	Hour of day

### Related Information

Intrinsics        None

Commands        None

Manuals          *Getting System Information Programmer's Guide (32650-90018)*

---

## CLOSELOG

NM and CM callable.

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

### Syntax

I32    I16    I16
CLOSELOG( <i>index,mode,logstatus</i> );

### Parameters

*index*      **32-bit signed integer by reference (required)**

Passes access capabilities to the user logging system. This is the value returned in the *index* parameter of the OPENLOG intrinsic.

*mode*        **16-bit signed integer by reference (required)**

Passes a value, indicating whether the user logging facility should suspend the process if it cannot complete the request for service immediately:

Value	Meaning
0	Wait
1	Nowait

*logstatus*    **16-bit signed integer by reference (required)**

Returns one of the following values, indicating the success/failure of the intrinsic call:

Value	Meaning
0	Successful
1	Nowait mode requested; user logging process busy
4	Incorrect <i>index</i> parameter passed
5	Incorrect <i>mode</i> parameter passed
7	Illegal capability; user logging (LG) or system supervisor (OP) capabilities required
9	Error occurred while writing
12	System out of disk space; user logging cannot proceed
14	Invalid access to user logging file
15	End-of-file encountered

**Operation Notes**

The number of users and log entries are independent of the number of times the `OPENLOG/CLOSELOG` intrinsics are called within an application. The user logging facility establishes a logging buffer (`LOGBUFF`) entry and increments the user count only at the first `OPENLOG` call. Another counter keeps track of the number of times `OPENLOG` and `CLOSELOG` are called. The counter is incremented for every `OPENLOG` and decremented for every `CLOSELOG`. This ensures that the entry in `LOGBUFF` is released only for the last `CLOSELOG` call (call counter = 0).

**Related Information**

Intrinsics      `OPENLOG`

Commands      None

Manuals      *User Logging Programmer's Guide (32650-90027)*

---

## COMMAND

NM and CM callable.

Programmatically executes a command.

### Syntax

CA	I16	I16
COMMAND( <i>cmdimage</i> , <i>cmderror</i> , <i>parmnum</i> );		

### Parameters

*cmdimage*     **character array (required)**

Passes an ASCII string of <=279 characters consisting of a command and parameters, terminated by a carriage return. The carriage return character (%15) must be the last character of the command string. Do not include a prompt character in this string.

*cmderror*     **16-bit signed integer by reference (required)**

Returns any error code set by the command:

- If *cmderror* is zero, no error occurred.
- If *cmderror* is negative, a warning was detected.

This the same error that would be returned by the command interpreter if *cmdimage* were executed interactively or in batch. The error message can be found in message set two of CATALOG.PUB.SYS.

---

**Note**     When the CI displays multiple errors for a single command, *cmderror* contains only the last error number displayed.

---

---

**Note**     Remote commands executed through the COMMAND or HPCICOMMAND intrinsics may not return a meaningful error status.

---

*parmnum*     **16-bit signed integer by reference (required)**

Returns a number indicating the error:

- If *parmnum* is positive, it is a file system error number and *cmderror* is non-zero.
- If *parmnum* is negative, it is the column number where the error occurred and *cmderror* is non-zero. (The command interpreter (CI) prints a caret there.)
- If *parmnum* is zero, then no additional error-related data is available.

---

**Note** The *cmderror* is the correct parameter to check to verify if *cmdimage* succeeded. The *parmnum* may provide additional information.

---

### Operation Notes

User-defined commands (UDCs) cannot be executed with the COMMAND intrinsic. RUN and other process creation commands are allowed if you, or the program calling COMMAND, have process handling (PH) capability. Command files and the implied RUN are not allowed.

The *cmdimage* parameter can contain any command except the following:

ABORT	JOB
BYE	LISTREDO
CHGROUP	MOUNT
DATA	MRJE
DISMOUNT	NRJE
DO	REDO
EOD	RESUME
EOJ	RJE
EXIT	SETCATALOG
HELLO	VSUSER

---

**Note** CIERROR 9103 is returned if *cmdimage* contains a command that is not supported.

---

For compatibility with MPE V/E, the COMMAND intrinsic converts three error numbers into CIERROR 975. They are: 9086, 9090, and 9091.

### Condition Codes

CCE (2) Request granted. Although, a CI warning may have been detected.

CCG (0) Request denied. An executor-dependent error, such as an erroneous parameter, prevented execution of the command. The numeric error code is contained in *cmderror*. The command with the error terminated before executing completely.

CCL (1) Request denied. The command was undefined.

### Related Information

Intrinsics	HPCICOMMAND
Commands	None
Manuals	<i>Command Interpreter Access and Variables Programmer's Guide</i> (32650-90011) and <i>MPE/iX Commands Reference Manual Volumes 1 and 2</i> (32650-90003 and 32650-90364)

---

## CREATE

NM and CM callable (differences noted below).

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

### Syntax

```
CREATE(formaldesig, entryname, pin, parm, loadflags,  
      stacksize, dlsiz, maxdata, priorityclass, rank);
```

### Parameters

*formaldesig* **character array (required)**

The name of the program to be created, specified either in MPE or HFS syntax. If you are using MPE syntax, *formaldesig* passes the name, group (optional), and account (optional) of the file containing the program to be run, and, if applicable, a lockword. If you are using HFS syntax, *formaldesig* passes the relative pathname or the absolute pathname of the file containing the program to be run. The last element of the array must be a blank. This parameter is equivalent to the *progfile* parameter of the RUN command.

*entryname* **character array (optional)**

Passes the declared entry point (label) in the program where execution is to begin when the process is activated. The last element of the array must be a blank. Specify the program's primary entry point with an array consisting of one blank character. (Equivalent to the *entrypoint* parameter of the RUN command.)

Default: The program's primary entry point is used.

*pin* **16-bit signed integer by reference (required)**

Returns the process identification number (PIN) of the newly created process. This PIN is used by other intrinsics to reference the new process. If an error is detected, a value of zero is returned.

*parm* **16-bit signed integer by value (optional)**

Passes information to the new process. The new process can access this information by using the GETINFO intrinsic. (Equivalent to the *parm=* parameter of the RUN command.)

Default: *parm*=0.

*loadflags***16-bit unsigned integer by value (optional)**

Specifies the loading options as follows:

<b>Bits</b>	<b>Value/Meaning</b>
15:1	Active: <ul style="list-style-type: none"> <li>0 Calling process not activated when the new process terminates</li> <li>1 The system reactivates the calling process (parent) when the new process terminates</li> </ul> Default: 0
14:1	Loadmap (equivalent to the <b>L</b> MAP parameter of the <b>R</b> UN command): <ul style="list-style-type: none"> <li>0 No map produced</li> <li>1 A list of the allocated (loaded) program is produced on the job/session list device; displays information about all SOMs loaded during process creation</li> </ul> Default: 0
13:1	Debug; ignored if nonprivileged and the new process requires privileged mode or if there is no read/write access for the new process program file (equivalent to the <b>D</b> EBUG parameter of the <b>R</b> UN command.): <ul style="list-style-type: none"> <li>0 Breakpoint not set</li> <li>1 Debug called prior to executing the first instruction of the new process</li> </ul> Default: 0
12:1	Privilege (equivalent to the <b>N</b> OPRIV parameter of the <b>R</b> UN command): <ul style="list-style-type: none"> <li>0 Program loaded in the mode specified when the program was linked</li> <li>1 Program loaded in nonprivileged mode</li> </ul> Default: 0

## CREATE

Bits	Value/Meaning
10:2	<p>LIBSEARCH bits; specifies a failed load if the <i>formaldesignator</i> parameter contains a program name which cannot be expressed using the MPE syntax. This option is equivalent to the <b>LIB=</b> parameter of the <b>RUN</b> command:</p> <p>00 (NM) If <i>formaldesig</i> was linked using the <b>XL=</b> parameter of the <b>LINK</b> command, then search the libraries specifying <b>XL=</b> first. Otherwise, search only the system libraries.</p> <p>(CM) Search only the system libraries (<b>SL.PUB.SYS</b>).</p> <p>01 (NM) Search the account public library (<b>XL.PUB.accountname</b>) in the account where <i>formaldesig</i> is located, then search the system libraries.</p> <p>(CM) Search the account public library (<b>SL.PUB.accountname</b>) in the account where <i>formaldesig</i> is located, then search the system libraries.</p> <p>10 (NM) Search the group library (<b>XL.groupname.accountname</b>) in the account where <i>formaldesig</i> is located, then the account public library, and then the system libraries.</p> <p>(CM) Search the group library (<b>SL.groupname.accountname</b>) in the account where <i>formaldesig</i> is located, then the account public library, and then the system libraries.</p> <p>Default: 00</p>
9:1	<p>Control Block; controls where the control blocks of device files are established and files of type message, RIO, and circular (equivalent to the <b>NOCB</b> parameter of the <b>RUN</b> command):</p> <p>0 Establish file control blocks in the process control block extension (PCBX) area of the CM stack</p> <p>1 Establish file control blocks in an extra data segment</p> <p>Default: 0</p>



Bits	Value/Meaning
7:2	Reserved for operating system
5:2	Stack dump; control the enabling/disabling mechanism where the stack is dumped in the event of an abort:  00 Stack dump mechanism enabled only at parent level 01 Stack dump mechanism enabled unconditionally 10 Stack dump mechanism enabled only at parent level 11 Stack dump mechanism disabled unconditionally for new process  Default: 00
4:1	Reserved for operating system
3:1	DL to initial Q; ignored if <i>formaldesig</i> specifies a native mode program file. Used for backward-compatibility with MPE V/E-based systems only.
0:3	Reserved for operating system; used only when bits (5:2)=01.

*stacksize*

**16-bit signed integer by value (optional)**

(NM) Ignored if *formaldesig* specifies an NM program file (used for backward-compatibility with MPE V/E-based systems only). The NM stack is created using NM default values. The CM stack is created using MPE V/E mode default values. (Equivalent to the `stack=` parameter of the `RUN` command.)

(CM) Passes the number of words assigned to the local stack area bounded by the initial Q and Z registers. A value of -1 indicates that the Segmenter will assign the default values (equivalent to omitting this parameter).

## CREATE

*dlsize*

### **16-bit signed integer by value (optional)**

(NM) Ignored if *formaldesig* specifies an NM program file (used for backward-compatibility with MPE V/E-based systems only). The NM stack is created using NM default values and the CM stack is created using MPE V/E maximum default values. (Equivalent to the **DL=** parameter of the **RUN** command.)

(CM) Passes the number of words in the user stack area bounded by the DL and DB registers. A value of -1 indicates that the Segmenter will assign the default values (equivalent to omitting this parameter).

*maxdata*

### **16-bit signed integer by value (optional)**

(NM) Ignored if *formaldesig* specifies a native mode program file (used for backward-compatibility with MPE V/E-based systems only). The NM stack is created using NM default values and the CM stack is created using MPE V/E maximum default values. (Equivalent to the **MAXDATA=** parameter of the **RUN** command.)

(CM) Passes the maximum size allowed for the process stack in words. This value overrides the value specified at program preparation. A value of -1 indicates that the Segmenter will assign the default value. If not specified in either the intrinsic call or the program file, the stack size remains the same.

*priorityclass*    **16-bit unsigned integer by value (optional)**

Returns the integer equivalent of two ASCII characters specifying the priority class where the new process is scheduled (AS, BS, CS, DS, ES). It is computed as:

$$\begin{aligned} & (\text{ASCII of first character} * 256) + \\ & (\text{ASCII of second character}) \end{aligned}$$

The integer equivalents are:

AS = 16,723  
BS = 16,979  
CS = 17,235  
DS = 17,491  
ES = 17,747

If a user is nonprivileged, the processes can be rescheduled into any of the five subqueues except the AS queue. This function is limited by the maximum priority assigned to the account by the system manager.

If a user is privileged, the processes can be scheduled into any subqueue, including the AS. A process in the AS or BS linear queue does not give up CPU voluntarily; it could loop infinitely and prevent other processes from accessing the CPU. (Equivalent to the **PRI=** parameter of the **RUN** command.)

Default: The calling process priority.

*rank*    **16-bit signed integer by value (optional)**

For backward-compatibility with MPE V/E-based systems only.

## CREATE

### Operation Notes

The **CREATE** intrinsic does the following:

1. Loads the program to be run by the new process into virtual memory.
2. Creates the process as the child of the calling process.
3. Initializes the process data areas.
4. Schedules the process.
5. Returns the new process identification number (PIN) to the requesting process.

The process is not created and the *pin* parameter is returned with a value of zero if one of the following conditions exist:

- The value (a nonexistent subqueue) specified is illegal for the *priorityclass* parameter.
- The *formaldesig* is illegal.
- The *entryname* is illegal.
- The program specified by *formaldesig* contains more capabilities than allowed. PM, MR, or DS capability is not allowed if the program name uses HFS syntax.
- The program specified by *formaldesig* uses HFS syntax and has PH capability, but the user does not have PH capability.
- A CM program cannot be loaded from the HFS directory.

The process is not created and the *pin* parameter is returned unmodified if one of the following conditions exist:

- The program file of the creating process does not have process handling (PH) capability.
- A required parameter (*formaldesig* or *pin*) is omitted.
- A reference parameter is not within the required range.
- The program name is equal to blank spaces.

### Condition Codes

CCE (2)	Request granted. The new process has been created.
CCG (0)	Request granted.
CCL (1)	Request denied. The specified <i>formaldesig</i> or <i>entryname</i> does not exist.

### Related Information

Intrinsics	ACTIVATE, SUSPEND
Commands	RUN
Manuals	<i>Process Management Programmer's Guide</i> (32650-90023)

## CREATEPROCESS

NM and CM callable (differences noted below).

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

### Syntax

```

      I*      I16      CA
CREATEPROCESS(createstatus,pin,formaldesig,
      I*A      I*A
      itemnums,items);
```

### Parameters

*createstatus* (NM) **32-bit signed integer by reference (required)**  
 (CM) **16-bit signed integer by reference (required)**

Returns a value indicating the type of success or failure. A negative value indicates that the associated message is a warning, an error terminates the intrinsic call, and a warning means the call continues. Check the job control word (JCW) before continuing. The following is a list of the possible values returned:

Value	Meaning
0	Successful
1	Process handling (PH) capability required
2	<i>Pin</i> or <i>formaldesig</i> parameter missing; or one of <i>itemnum</i> pair is missing (for example, 11 and 12 or 19 and 24)
3	Parameter address (other than <i>createstatus</i> ) out of bounds
4	Out of system resources
5	Process not created; invalid <i>itemnum</i> specified; or, for a compatibility mode process, total requested stack space exceeds maximum configured.
6	Process not created; <i>formaldesig</i> does not exist
7	Process not created; <i>formaldesig</i> invalid
8	Process not created; entry name invalid or does not exist
-9	Process created; LIBSEARCH bits ignored
-10	Process created; <i>itemnum</i> =19 ignored

## CREATEPROCESS

Value	Meaning
15	Process not created; reserved <i>item</i> was specified
16	Process not created; hard load error occurred (for example, I/O error reading <i>formaldesig</i> )
17	Process not created; illegal value specified for <i>itemnum</i> =7
18	Process not created; specified <b>\$STDIN</b> could not be opened
19	Process not created; specified <b>\$STDLIST</b> could not be opened
20	Process not created; string to be passed to new process invalid (pointer without length, length without pointer, or length exceeds stack size of calling process)

***pin***            **16-bit signed integer by reference (required)**

Returns the process identification number (PIN) of the newly created process. This PIN is used by other intrinsics to reference the new process. If an error is detected, a value of zero is returned.

***formaldesig***    **character array (required)**

Passes the name of the program file to be run. Terminate the string with any nonalphanumeric character other than a period (.) or a backslash (/). This parameter is equivalent to the *progfile* parameter of the RUN command.)

***itemnums***        (NM) **32-bit signed integer array (optional)**

(CM) **16-bit signed integer array (optional)**

Specifies which *item* value is to be passed. (Refer to Table 4-1.)

Values passed may be passed by value.

***items***            (NM) **32-bit signed integer array (optional)**

(CM) **16-bit signed integer array (optional)**

Specifies if the *formaldesig* parameter contains a program name that is unrecognizable using MPE syntax, LIBSEARCH will result in a failed load.

Table 4-1. CREATEPROCESS Itemnum/Item Values

Itemnum	Item Description
0	Indicates the end of the option list. No value is required in the corresponding <i>item</i> .
1	The address must be a byte address (type-coerced) containing the entry point name in the program where the new process is to begin execution (equivalent to the <i>entrypoint</i> parameter of the <b>RUN</b> command.) The name consists of a string of characters terminated by a blank.
2	A value passed to the new process (equivalent to the <b>PARM=</b> parameter of the <b>RUN</b> command.)
3	<p>The load options for loading the program file of the new process, designated as:</p> <p>(NM)</p> <p>Bits (0:16)        Zeros Bits (16:16)      Load options</p> <p>(CM)</p> <p>Bits (0:16)        Load options</p> <p>Item 3 has the same definition as the <i>loadflag</i> parameter of the <b>CREATE</b> intrinsic. If both <i>itemnums</i> 3 and 19 are specified, the <b>LIBSEARCH</b> option is ignored or if <b>LIBSEARCH</b> is set to a value other than its default (00), a warning is returned indicating the <b>LIBSEARCH</b> option was ignored.</p>
4,5,6	(CM Only) Ignored if <i>formaldesig</i> specifies an NM program file; created using MPE V/E maximum default values (equivalent to the <b>STACK=</b> , <b>DL=</b> , and <b>MAXDATA=</b> parameters of the <b>RUN</b> command).
7	<p>The priority class where the new process is scheduled (AS, BS, CS, DS, ES); is computed as (equivalent to the <b>PRI=</b> parameter of the <b>RUN</b> command.):</p> <p style="text-align: center;">(ASCII of first character * 256) + (ASCII of second character)</p> <p>The integer equivalents are:</p> <p>AS = 16,723 BS = 16,979 CS = 17,235 DS = 17,491 ES = 17,747</p> <p>If in user mode (nonprivileged), any priority class can be specified except AS; this is limited by the maximum priority assigned to the account by the system manager. If PM capability has been assigned, processes can be scheduled into all subqueues. A process in the linear queue AS or BS does not give up the CPU voluntarily, it could loop infinitely and prevent other processes from accessing the CPU.</p> <p>Default: The calling process priority class.</p>

## CREATEPROCESS

**Table 4-1. CREATEPROCESS Itemnum/Item Values (continued)**

Itemnum	Item Description
8	The address must be a byte address (type-coerced) containing a definition of the file to be used as <code>\$\$STDIN</code> for the new process (equivalent to the <code>STDIN=</code> parameter of the <code>RUN</code> command). Must contain an ASCII string (terminated by a carriage return) specifying the file to be used as <code>\$\$STDIN</code> (that is, everything after the <code>FILE formaldesig=;</code> portion of the file equation). If <code>itemnum=8</code> is not specified, the default <code>\$\$STDIN</code> and <code>\$\$STDLIST</code> (current for the creating (parent) process) are used for the new process creation.
9	The address must be a byte address (type-coerced) containing a definition of the file to be used as <code>\$\$STDLIST</code> for the new process (equivalent to the <code>STDLIST=</code> parameter of the <code>RUN</code> command). If <code>itemnum=9</code> is not specified, the default <code>\$\$STDIN</code> and <code>\$\$STDLIST</code> (current for the creating (parent) process) are used for the new process creation.
10	Specifying this item is equivalent to calling the <code>ACTIVATE</code> intrinsic with the newly created process. The value passed is equivalent to the <code>allow</code> parameter of <code>ACTIVATE</code> , except bit (31:1) is the parent activation bit and bit (30:1) is the child activation bit.
11	The address must be a byte address (type-coerced) containing information to be passed to the new process. The length of the string must be specified in <code>itemnum=12</code> (equivalent to the <code>INFO=</code> parameter of the <code>RUN</code> command), accessible through the <code>GETINFO</code> intrinsic.
12	The length (in bytes) of the string referenced by <code>itemnum=11</code> . The maximum allowable value of the <code>INFO</code> string length is 1024 bytes. Accessible through the <code>GETINFO</code> intrinsic.
13	Not used.
14	The address must be a byte address (type-coerced) containing a definition of the file to be used as <code>\$\$STDERR</code> for the new process. If <code>itemnum=14</code> is not specified, the <code>\$\$STDERR</code> (current for the creating (parent) process) is used for the new process creation.  Redirecting output to <code>\$\$STDERR</code> permits the user to write errors to a user-specified disc file.
15-18	(NM Only) Reserved for the system.
19	(NM Only) The address (type-coerced) containing a list of unresolved external references found in <code>formaldesig</code> (equivalent to the <code>XL=</code> parameter of the <code>RUN</code> command). When <code>formaldesig</code> contains a name in MPE syntax, the file names must follow file naming conventions and be separated by commas. When <code>formaldesig</code> contains a name in HFS syntax, you must use an absolute pathname and the file names specified in this item must be fully qualified.  If <code>itemnums 3</code> and <code>19</code> are specified, the <code>LIBSEARCH</code> option in <code>itemnum=3</code> is ignored or if <code>LIBSEARCH</code> is set to a value other than its default (00), a warning is returned indicating the <code>LIBSEARCH</code> option was ignored. The length of the string must be specified in <code>itemnum=24</code> .
20-22	(NM Only) Reserved for operating system.
23	(NM Only) The address (type-coerced) containing the name of a procedure to link unsatisfied references to. The name consists of a string of characters terminated by an ASCII carriage return character or a blank (equivalent to the <code>UNSAT=</code> parameter of the <code>RUN</code> command).



Table 4-1. CREATEPROCESS Itemnum/Item Values (continued)

Itemnum	Item Description
24	(NM Only) The length of the string referenced by <i>itemnum</i> =19.
25	(NM Only) Reserved for operating system.
26	The address must be a byte address containing the maximum size (in bytes) that the NM stack can grow (equivalent to the <b>NMSTACK=</b> parameter of the <b>RUN</b> command).
27	The address must be a byte address containing the maximum size (in bytes) that the NM heap can grow (equivalent to the <b>NMHEAP=</b> parameter of the <b>RUN</b> command).

### Condition Codes

- CCE (2) Request granted.
- CCG (0) Request granted. Check the *createstatus* parameter to see if warning codes were returned. \CCL (1)\Request denied. Check the *createstatus* parameter to determine why the request was denied.

### Related Information

- Intrinsics CREATE
- Commands RUN
- Manuals *Process Management Programmer's Guide* (32650-90023)

---

## 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).

### Syntax

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

### Parameters

*transcode*    **16-bit signed integer by value (required)**

Passes one of the following translation identifiers:

Value	Meaning
0	User-supplied table specified in the <i>transtable</i> parameter
1	Convert EBCDIC to ASCII
2	Convert ASCII to EBCDIC
3	Reserved for operating system
4	Reserved for operating system
5	Convert EBCDIK to KANA8 (JIS)
6	Convert KANA8 (JIS) to EBCDIK

*inbuffer*    **character array (required)**

Passes the string of characters to be translated.

*outbuffer*    **character array (optional)**

Passes the translated character string. If an *outbuffer* parameter is not specified, all translation occurs within *inbuffer*. The *inbuffer* and *outbuffer* parameters can specify the same array.

*bufferlength*    **16-bit signed integer by value (required)**

Passes the number of characters to be converted.

*transtable*      **character array (required when *transcode*=0, otherwise optional)**

Passes the user-defined translation table. The contents and order of *transtable* define the translation process. *Transtable* must be  $\leq 256$  bytes. *Transtable* is constructed so that each byte corresponds to a byte value in the source string to be translated. For example, the first byte in *transtable* gives the code to be substituted for source bytes whose value is 0.

### Condition Codes

CCE (2)      Request granted. Translation performed successfully.

CCG (0)      Not returned.

CCL (1)      Request denied. An error occurred.

### Related Information

Intrinsics      None

Commands      None

Manuals      *Data Types Conversion Programmer's Guide (32650-90015)*

**DASCII**

NM and CM callable.

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

**Syntax**

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

**Functional Return**

*numchar*      **16-bit signed integer (assigned functional return)**

Returns the number of characters in the resulting string.

**Parameters**

*binvalue*      **32-bit signed integer by value (required)**

Contains the binary number to be converted to an ASCII string.

*base*            **16-bit signed integer by value (required)**

Contains one of the following translation identifiers:

Value	Meaning
8	Convert to octal (pad with zeros)
10	Convert to decimal (left-justified)
-10	Convert to decimal (right-justified)
16	Convert to hexadecimal (pad with zeros)

If any other number is entered in this parameter, the intrinsic causes the process to abort.

*asciieqv*      **character array (required)**

Returns the converted value. Must be long enough to contain the result (<= 11 characters).

- For octal conversions (*base*=8), 11 characters (including leading zeros) are returned; *numchar* returns the number of significant (right-justified) characters (excluding leading zeros). If *binvalue*=0, the length returned is 1.
- For decimal conversions (*base*=10), *binvalue* is considered a 32-bit two's complement integer ranging from -2,147,483,648 to +2,147,483,647. If *binvalue*=0, only one zero character is returned in *asciieqv*; *numchar*

returns the total number of characters (including the sign). For example, if *binvalue*=0, the length returned is 1.

- For hexadecimal conversions (*base*=16), 8 characters (including leading zeros) are returned. The digits can be 0..9 and A..F. *Numchar* returns the number of significant (right-justified) characters (excluding leading zeros). For example, if *binvalue*=32, *numchar* is 2 and *asciiqv* will be 00000020.

### Related Information

Intrinsics	ASCII
Commands	None
Manuals	<i>Data Types Conversion Programmer's Guide</i> (32650-90015)

---

## DATELINE

NM and CM callable.

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

### Syntax

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

### Parameters

*datebuffer* character array (required)

Returns the date and time information, in a 27-character array, in the format:

```
FRI, MAY 19, 1989, 12:06 PM
```

### Related Information

Intrinsics ALMANAC

Commands None

Manuals *Getting System Information Programmer's Guide* (32650-90018)

## 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.

### Syntax

I32	CA	I16V
<i>dbineqv</i> :=DBINARY( <i>dascieqv</i> , <i>length</i> );		

### Functional Return

*dbineqv*      **32-bit signed integer (assigned functional return)**

Returns the converted 32-bit binary value.

### Parameters

*dascieqv*      **character array (required)**

Passes the octal or signed decimal number (in ASCII characters) to be converted. If the character string begins with a percent sign (%), it is treated as an octal value. If the character string begins with a dollar sign (\$), it is treated as a hexadecimal value. In this case, the string must be less than 6 characters and only 0 through 9, a through f, and A through F are allowed. If the string begins with a plus sign, minus sign, or a number, it is treated as a decimal value. Leading blanks are not allowed and are treated as illegal characters.

*length*      **16-bit signed integer by value (required)**

Returns the length (number of bytes) of the ASCII-coded value:

- If the value of *length* is 0, the intrinsic returns 0 to the calling process.
- If the value of *length* is negative, the intrinsic causes the process to abort.

### Condition Codes

- CCE (2)      Request granted. A 32-bit binary value is returned to the process.
- CCG (0)      Request denied. A word overflow, possibly resulting from too many characters (*dascieqv* number too large), occurred in the value (*dbineqv*) returned.
- CCL (1)      Request denied. An illegal character was encountered in *dascieqv*. For example, the digits 8 or 9 were specified in an octal value.

## **DBINARY**

### **Related Information**

Intrinsics      **BINARY**

Commands      None

Manuals      *Data Types Conversion Programmer's Guide (32650-90015)*



---

## DEBUG

NM and CM callable.

Invokes the debug facility from an interactive program and allows object code to be analyzed.

---

**Note** Consult the *MPE/iX System Debug Reference Manual* (32650-90013) before attempting to use the debug facility.

---

### Syntax

```
DEBUG;
```

### Operation Notes

The debug facility has its own set of commands and a symbolic window display feature. The intrinsic call acts as a hard-coded breakpoint. Execution of the calling program is halted and the debug facility prompt is displayed. If the call is made from a batch program, it is ignored.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>MPE/iX System Debug Reference Manual</i> (32650-90013)

---

## 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.

### Syntax

I16	I16V
<i>dldbsize</i> :=DLSIZE( <i>size</i> );	

### Functional Return

*dldbsize*      **16-bit signed integer (assigned functional return)**

Returns the granted number of half words for the DL-to-DB area. This value is negative, unless error condition CCL (1) is returned; a positive value is possible.

### Parameters

*size*            **16-bit signed integer by value (required)**

Passes the desired number of half words for the DL-to-DB area, represented by a negative value. A *size* of 0 is permitted and resets the DL-to-DB area of the CM stack to the original value assigned when the process was created (initial DL). The size granted is the absolute value of *size*, rounded up so that the distance between the beginning of the segment and DB is a multiple of 128 half words.

### Operation Notes

If expanding, all information in the area between DL and DB is saved. If contracting, all information in the area between DL and DB is lost.

If the DL area has been expanded, it can be contracted by calling **DLSIZE** with a *size* parameter requesting a smaller DL-to-DB area.

When called from programs running in NM the **DLSIZE** intrinsic affects the CM stack only; programmatic expansion and contraction in NM is not necessary.

---

### Caution

Because of differences between DB-relative addressing (used in compatibility mode) and virtual addressing (used in native mode), calls to the **DLSIZE** intrinsic in mixed mode programming may cause stack corruption.

Hewlett-Packard does not recommend using the **DLSIZE** intrinsic in programs that move between Native mode and compatibility mode.

---

**Condition Codes**

- CCE (2) Request granted. The value returned is  $\geq$  the value requested.
- CCG (0) Request denied. The requested size exceeded the maximum limit allowed. The maximum limit allowable is granted, and its size is returned.
- CCL (1) Request denied. An illegal *size* parameter was specified, the size requested is less than the initial DL size, or the *size* parameter was a positive integer. The original area size, assigned when the CM stack segment was created, is granted.

**Related Information**

- Intrinsics None
- Commands None
- Manuals *Introduction to MPE XL for MPE V Programmers (30367-90005)*

---

## DMOVIN

NM and CM callable.

Copies data from an extra data segment into a data area. Data segment management (DS) capability is required.

---

**Note** 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.

---

### Syntax

U16V    I16V    I16V    U16A
DMOVIN( <i>index,displacement,number,location</i> );

### Parameters

- index*      **16-bit unsigned integer by value (required)**  
Specifies the index of the extra data segment, obtained from a GETDSEG intrinsic call.
- displacement*      **16-bit signed integer by value (required)**  
Specifies the displacement, in half words, of the first half word in the string to be transferred from the first half word in the data segment. This value must be  $\geq 0$ .
- number*      **16-bit signed integer by value (required)**  
Specifies the size, in half words, of the data string to be transferred. This value must be  $\geq 0$ .
- location*      **16-bit unsigned integer array (required)**  
Contains the array (buffer) in the stack where the data string is to be moved.

### Operation Notes

When DMOVIN is called, a bounds check is initiated to ensure that the data is taken from within the data segment boundaries and moved to an available data area. For example, in Figure 4-1, to move four half words from locations 422 through 425 of the extra data segment (whose *index* is 21) to array ARA (located in your data area), the intrinsic call would be:

```
DMOVIN(21,422,4,ARA[10]);
```

The *index* is 21 (refer to GETDSEG); *displacement* within the data segment is 422; the *number* of half words to move into the data area is 4; and the *location* to begin transferring the data is the address of ARA[10]. If ARA is an array of half words, the four half words are moved to ARA[10] through ARA[13], as shown in Figure 4-1.

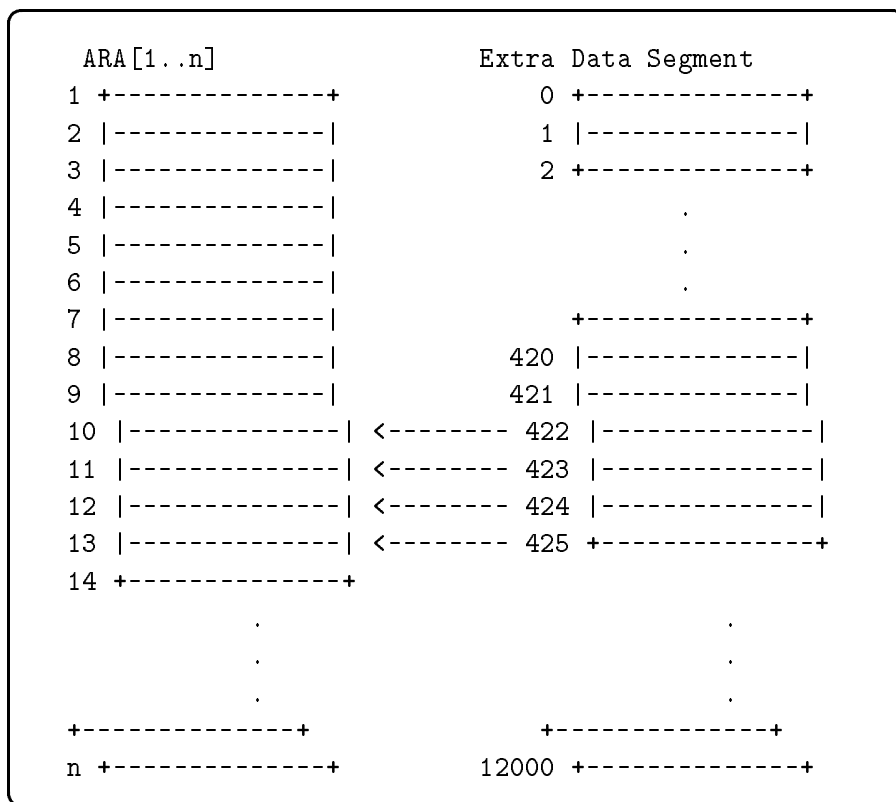


Figure 4-1. DMOVIN Data Movement.

### Condition Codes

- CCE (2) Request granted.
- CCG (0) Request denied. A bounds-check failure occurred.
- CCL (1) Request denied. Illegal *index* or *number* parameter.

## **DMOVIN**

### **Related Information**

Intrinsics      DMOVOUT, GETDSEG

Commands      None

Manuals      *Introduction to MPE XL for MPE V Programmers (30367-90005)*

---

## DMOVOUT

NM and CM callable.

Copies data from the data area to an extra data segment. Data segment management (DS) capability is required.

---

**Note** 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.

---

### Syntax

U16V	I16V	I16V	U16A
DMOVOUT( <i>index,displacement,number,location</i> );			

### Parameters

- index*      **16-bit unsigned integer by value (required)**  
 Specifies the index of the extra data segment, obtained from a GETDSEG intrinsic call.
- displacement*      **16-bit signed integer by value (required)**  
 Specifies the displacement, in half words, of the first half word in the string to be transferred from the first half word in the data segment. This value must be  $\geq 0$ .
- number*      **16-bit signed integer by value (required)**  
 Specifies the size, in half words, of the data string to be passed. This value must be  $\geq 0$ .
- location*      **16-bit unsigned integer array (required)**  
 Passes the data string to be moved to the extra data segment.

## DMOVOUT

### Operation Notes

When DMOVOUT is called, a bounds check is initiated to ensure that the data is taken from within an available data area and moved to an area within the extra data segment boundaries. For example, in Figure 4-2, to move four half words from ARA[10] to the data segment whose *index* is 2 (from a GETDSEG call), starting at location 201 within the segment, the intrinsic call could be:

```
DMOVOUT (2,201,4,ARA[10]);
```

The *index* is 2; the *displacement* within the data segment is 201; the *number* of half words to be moved to the data segment is 4; and the starting *location* of the data is the address of ARA[10]. If ARA is an array of half words, the four half words, from ARA[10] to ARA[13], are moved to half word offset 201 through 204 of the data segment, as shown in Figure 4-2.

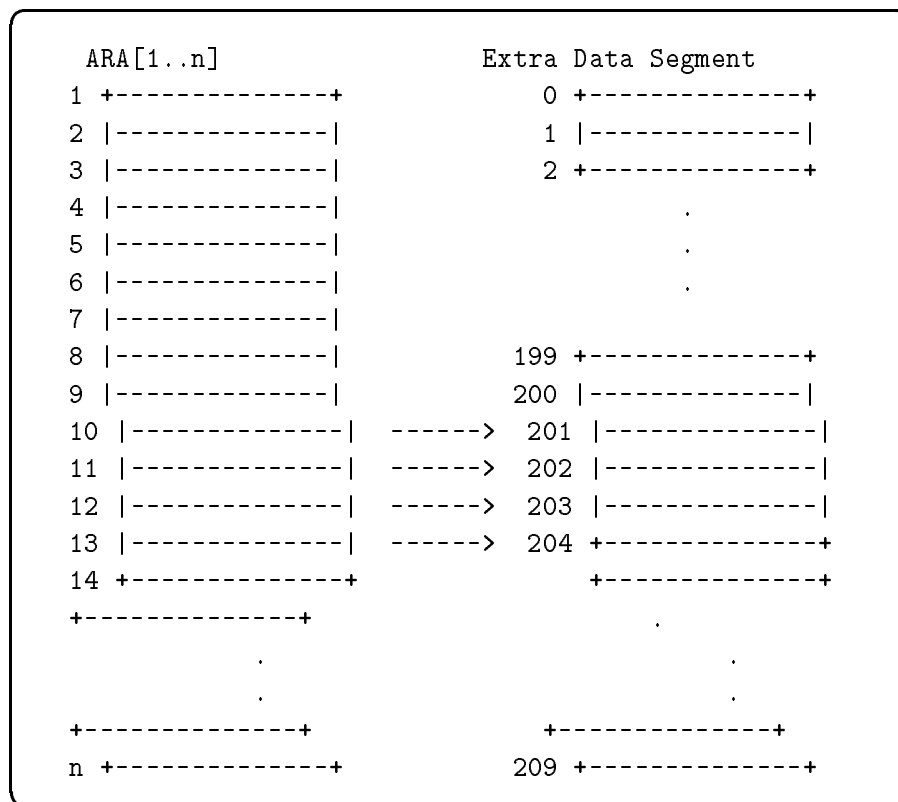


Figure 4-2. DMOVOUT Data Movement.

### Condition Codes

- CCE (2) Request granted.
- CCG (0) Request denied. Out of bounds-check failure.
- CCL (1) Request denied. Illegal *index* or *number* parameter.



**Related Information**

Intrinsics      DMOVIN

Commands      None

Manuals      *Introduction to MPE XL for MPE V Programmers (30367-90005)*

---

## 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.

### Syntax

I32 U16A I16 I16 I16
ENDLOG( <i>index,data,length,mode,logstatus</i> );

### Parameters

- index*      **32-bit signed integer by reference (required)**  
Passes the access to the user logging system. This is the value returned in the *index* parameter of the **OPENLOG** intrinsic.
- data*      **16-bit unsigned integer array (required)**  
Passes the information to be logged. A log record contains 128 half words, where 119 half words are available for writing data. For the most efficient use of log file space, structure arrays with lengths in multiples of 119 half words.
- length*     **16-bit signed integer by reference (required)**  
Passes the length of the data in *data*. A positive value indicates half words, while a negative value indicates bytes. If the length is greater than 119 half words (or 238 bytes), the information in *data* is divided into two or more physical log records.
- mode*      **16-bit signed integer by reference (required)**  
Passes a value indicating whether the logging process should suspend your process if it cannot complete your request for service immediately. If it is not possible to log the transaction and *mode* is set to 1 (nowait), the **ENDLOG** intrinsic indicates through *logstatus* that it could not complete your request.

Value	Meaning
0	Wait
1	Nowait

*logstatus*      **16-bit signed integer by reference (required)**

Returns one of the following values, indicating the success/failure of the intrinsic call:

Value	Meaning
0	Successful
1	Nowait mode requested, and the user logging process is busy
2	Parameter out of bounds
4	Incorrect <i>index</i> parameter passed
5	Incorrect <i>mode</i> parameter passed
9	Error occurred while writing
12	System out of disk space; user logging cannot proceed
14	Invalid access
15	End-of-file encountered

### Operation Notes

Use the *data* parameter of this intrinsic to post user data to the logging file. This use of ENDLOG is similar to using the WRITELOG intrinsic, but also denotes the end of a logical transaction.

### Related Information

Intrinsics	WRITELOG, OPENLOG
Commands	None
Manuals	<i>User Logging Programmer's Guide</i> (32650-90027)

---

## 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 *uslfnm*. The old USL file is copied to the new file with the same file name; the old USL file is then deleted.

---

**Note** A USL contains CM object code and is meaningful only in the CM program development process.

---

### Syntax

I16	I16V	I16V
<i>filenum</i> := EXPANDUSLF( <i>uslfnm</i> , <i>increment</i> );		

### Functional Return

*filenum*      **16-bit signed integer (optional)**

Returns the new file number. If an error occurs (condition code returns CCL (1)), an error number is returned. The condition code must be tested immediately upon return from this intrinsic. If an error number is used instead of a file number in the *uslfnm* parameter, unpredictable results occur. Following are the possible error numbers and their meanings:

Value	Meaning
0	File specified by <i>uslfnm</i> was empty, an unexpected end-of-file was encountered when reading the old USL file, or an unexpected end-of-file was encountered when writing the new <i>uslfnm</i>
1	Unexpected I/O error occurred; can occur on the old USL file, or the new <i>uslfnm</i> where the information is being copied to
7	Unable to open the new USL file
8	Unable to close (purge) the old USL file
9	Unable to close (save) the new USL file
10	Unable to close \$NEWPASS
11	Unable to open \$OLDPASS

**Parameters**

- uslfnm*      **16-bit signed integer by value (required)**  
Passes the file number of the file.
- increment*    **16-bit signed integer by value (required)**  
Passes the number of records the length of the USL file is to be changed:
- If a positive value, the new USL file is longer than the old USL.
  - If a negative value, the new USL file is shorter than the old USL.

**Condition Codes**

- CCE (2)      Request granted. The new file number is returned.
- CCG (0)      Not returned.
- CCL (1)      Request denied. An error number was returned to *filenum*.

**Related Information**

- Intrinsics    ADJUSTUSLF
- Commands     None
- Manuals      *MPE Segmenter Reference Manual* (30000-90011)

---

## FATHER

NM and CM callable.

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

### Syntax

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

### Functional Return

*pin*            **16-bit signed integer (assigned functional return)**  
Returns the process identification number (PIN) of the parent process.

### Condition Codes

CCE (2)        Request granted. The parent is a user process.  
CCG (0)        Request granted. The parent is a job or session main process (logon CI).  
CCL (1)        Request granted. The parent is a system process.

### Related Information

Intrinsics     None  
Commands      None  
Manuals        *Process Management Programmer's Guide (32650-90023)*

## 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.

### Syntax

I16V	I16	I16	I32	I16
FCHECK( <i>filenum</i> , <i>fserrorcode</i> , <i>translog</i> , <i>blocknum</i> , <i>numrecs</i> );				

### Parameters

- |                    |  |
|--------------------|--|
| <i>filenum</i>     | <b>16-bit signed integer by value (optional)</b>   |
|                    | Passes the file number of the file for which error information is to be returned. If <i>filenum</i> is set to zero or if it is not specified, error information is returned about the last failed FOPEN call.  |
| <i>fserrorcode</i> | <b>16-bit signed integer by reference (optional)</b>   |
|                    | Returns a file system error code indicating the type of error that occurred. Refer to Table 4-2.   |
| <i>translog</i>    | <b>16-bit signed integer by reference (optional)</b>   |
|                    | Returns the number of half words read or written if an I/O error occurred. (This value is recorded in the transmission log.)   |
| <i>blocknum</i>    | <b>32-bit signed integer by reference (optional)</b>   |
|                    | Returns the physical record count for a nonspoolfile or the logical record count for a spoolfile: <ul style="list-style-type: none"> <li>■ For fixed-length and undefined-length record files, the physical count is the number of physical records transferred to or from the file since FOPEN.</li> <li>■ For variable-length record files, the physical count is the last rewind, rewind/unload, space forward or backward to tape mark.</li> </ul> |
| <i>numrecs</i>     | <b>16-bit signed integer by reference (optional)</b>   |
|                    | Returns the number of logical records in the bad block (blocking factor). (ASC only) This parameter is not meaningful for asynchronous devices.  |

**FCHECK****Table 4-2. File System Error Codes**

<b>Error Code</b>	<b>Description</b>
0	Successful or end-of-file (EOF)
1	Illegal DB register setting
2	Illegal capability
3	Required parameter missing
4	Disk space allocation disabled on all domain disks
5	DRT number >511
6	Device has no spare blocks available
7	Unformatted or uninitialized media on device
8	Illegal parameter value
9	Invalid file type specified in <i>foption</i>
10	Invalid record size specification
11	Invalid resultant block size
12	Record number out of range
13	Cannot open file multiaccess
16	More than 255 opens of a file
15	Exceeded maximum file opens for write access
17	Magnetic tape runaway
18	Device powered up
19	Forms control was reset
20	Invalid operation requested (for example, terminal type set to -24 which is an invalid entry)
21	Data parity error
22	Timeout (read timeout value set with <b>FCONTROL</b> <i>itemnum</i> =4)
23	End-of-tape
24	Device not ready; not online, modem dsr signal not “high”, or line disconnected)
25	No write ring on tape
26	Transmission error
27	I/O timeout; block mode read timer expired (timer set by system, not user changeable)
28	Timing error; data arrived at controller faster than it could be serviced (data overrun)



Table 4-2. File System Error Codes (continued)

Error Code	Description
29	Start I/O (SIO) failure
30	Unit failure
31	End-of-line (EOL) special character terminator or additional end-of-record (AEOR) character (defined by <b>FCONTROL</b> <i>itemnum</i> =25)
32	Software abort of I/O operation (for example, an <b>ABORTIO</b> command was entered)
33	Data lost or no buffer available
34	Unit not online
35	Data set not ready
36	Invalid disk address
37	Invalid memory address
38	Tape parity error
39	Recovered tape error
40	Operation inconsistent with access type
41	Operation inconsistent with record type
42	Operation inconsistent with device type (for example, attempted to set an invalid line speed through <b>FCONTROL</b> <i>itemnum</i> =11)
43	Write exceeds record size
44	Update at record zero
45	Privileged file violation
46	Out of disk space
47	I/O error on a file label
48	Invalid operation; multiple file access
49	Unimplemented function
50	Nonexistent account
51	Nonexistent group
52	Nonexistent permanent file
53	Nonexistent temporary file
54	Invalid file reference

**FCHECK****Table 4-2. File System Error Codes (continued)**

<b>Error Code</b>	<b>Description</b>
55	Device unavailable
56	Invalid device specification
57	Out of virtual memory
58	No file passed
59	Standard label violation
60	Global RIN not available or insufficient system resources
61	Out of group disk space
62	Out of account disk space
63	No nonshareable device capability
64	Program not prepared with multiple RIN capability
66	Plotter limit switch reached
68	Insufficient system resources
69	I/O error
70	I/O error while printing header/trailer, or an <b>FOPEN</b> or <b>FCLOSE</b> failed
71	Too many files open
72	Invalid file number
73	Bounds violation
74	No room in stack for file entry
77	Nowait I/O operation pending
78	No nowait I/O pending for any file
79	No nowait I/O pending for any special file
80	Spoolfile size exceeds configuration
81	No spool class defined in system
82	Insufficient space in spool class for I/O request
83	I/O error on spoolfile
84	Device unavailable for spoolfile
85	Operation inconsistent with spooling (for example, attempt to read hardware status)
86	Spooling internal error
87	Bad spoolfile block

Table 4-2. File System Error Codes (continued)

Error Code	Description
88	Nonexistent spoolfile
89	Power failure
90	Exclusive violation; file being accessed
91	Exclusive violation; file accessed exclusively
92	Lockword violation
93	Security violation
94	User not creator
95	Read not completed; the terminal user pressed <b>(BREAK)</b> (The user must enter the <b>RESUME</b> command to continue. The file system sees the restart, the application does not.)
96	Disk I/O error
97	No control-Y PIN
98	Read timer overflow; read exceeded maximum capacity of read timer (655.35 seconds)
99	BOT and backspace tape
100	Duplicate permanent file name
101	Duplicate temporary file name
102	I/O error on directory
103	Permanent directory overflow
104	Temporary directory overflow
105	Bad variable block structure
106	Extent size exceeds maximum
107	Insufficient space for user labels
108	Invalid file label
109	Invalid carriage control
110	Attempt to save permanent file as temporary
111	No save files (SF) capability
112	No mountable volumes (UV) capability
113	Volume set not mounted; mount problem
114	Volume set not dismounted; dismount problem
115	Attempted rename across volume sets rejected

**FCHECK****Table 4-2. File System Error Codes (continued)**

Error Code	Description
116	Invalid tape label <b>FOPEN/HPFOPEN</b> parameters
117	Attempted write on an unexpired tape file
118	Invalid header or trailer tape label
119	I/O error positioning tape for tape labels
120	Attempted write on an IBM standard tape label
121	Tape label lockword violation
122	Tape label table overflow
123	End of tape volume set
124	Attempt to append labeled tape
125	Expiration date later than preceding file
126	Character set number must be between 0 and 31
127	Form number must be between 0 and 31
128	Logical page number must be between 0 and 31
129	Vertical format number must be between 0 and 31
130	Number of copies must be between 1 and 32,767
131	Number of overlays must be between 1 and 8
132	Page length parameter must be between 12 (=3) and 68 (=17)
133	Picture number must be between 0 and 31
134	Extended capability parameter must be 0 (OFF) or 1 (ON)
137	Defective track on foreign disk
138	Track does not exist on foreign disk
139	Deleted record on IBM diskette
148	Inactive RIO record
149	Missing item number or return variable
150	Invalid <i>itemnum</i>
151	Undefined file type
152	Unrecognized option in <b>FOPEN/HPFOPEN</b> <i>device</i> parameter
153	Expecting semicolon (;) or carriage return in <b>FOPEN/HPFOPEN</b> <i>device</i> parameter
154	Environment file open error
155	File not environment file; check file code or record size

Table 4-2. File System Error Codes (continued)

Error Code	Description
156	Header record incorrect
157	Uncompiled environment file
158	Error reading environment file
159	Error closing environment file
160	Error doing <b>FDEVICECONTROL</b> from environment file
161	Too many parameters in device string overflow
162	Expecting “=” after option in <i>device</i> parameter
163	Incorrect <b>ENV</b> backreference in file equation
164	<i>Device</i> parameter too large or missing carriage return
165	Invalid density specification
166	<b>FFILEINFO</b> failed while accessing remote spoolfile
167	Spoolfile label error; cannot insert environment file name
168	Item not supported on remote system
170	Record marked deleted; pointer positioned to a record marked for deletion
171	(KSAM) Duplicate key value not allowed
172	(KSAM) Key not found; invalid key value
173	(KSAM) <i>tcount</i> parameter larger than record size
174	(KSAM) Cannot get extra data segment
175	(KSAM) Internal KSAM error
176	(KSAM) Illegal extra data segment
177	(KSAM) Too many extra data segments for this process
178	(KSAM) Not enough virtual memory for extra data segment
179	(KSAM) File must be locked before issuing this intrinsic
180	(KSAM) KSAM file must be rebuilt; KSAM version does not handle a file built by previous version
181	(KSAM) Invalid key starting position
182	(KSAM) File empty
183	(KSAM) Record does not contain all keys
184	(KSAM) Invalid record number

**FCHECK****Table 4-2. File System Error Codes (continued)**

<b>Error Code</b>	<b>Description</b>
185	(KSAM) Sequence error in primary key
186	(KSAM) Invalid key length
187	(KSAM) Invalid key specification
188	(KSAM) Invalid device specification
189	(KSAM) Invalid record format
190	(KSAM) Invalid key blocking factor value
191	(KSAM) Record does not contain search key for deletion; specified key value points to record not containing that value
192	(KSAM) System failure occurred while KSAM file was opened
193	(KSAM) \$STDIN/\$STDLIST cannot be redirected to KSAM files
194	(KSAM) KSAM files not allowed for global AFTs
196	(KSAM) Language not supported
197	(KSAM) native language internal error
198	(KSAM) Invalid version number in KSAM file
201	(DS) Unable to access DSLINE table
204	(DS) Unable to allocate an extra data segment for DS/DSN 3000
205	(DS) Unable to expand the DS/DSN 3000 extra data segment
214	(DS) Requested DS line has not been opened with a DSLINE command
216	(DS) Message rejected by remote computer
217	(DS) Insufficient amount of user stack available
221	(DS) Internal DS error; invalid message format
224	File equations for a remote file constitute a loop
227	RFA/RDBA could not find or create a remote session
239	Communication interface error; trace malfunction
240	(DS) Local communication line not opened by operator

Table 4-2. File System Error Codes (continued)

Error Code	Description
241	(DS) DS line in use exclusively or by another subsystem
242	(DS) Internal DS software malfunction
243	(DS) Remote computer not responding
246	Communications interface error; remote computer disconnected
249	Communications interface error; unable to enter remote session
255	Communications interface error; connection lost
302	Invalid item number for <b>FDEVICECONTROL</b>
303	Invalid access for item number to <b>FDEVICECONTROL</b>
304	Attempt to change terminal parity in 8-bit mode
305	Invalid format in terminal configuration file
306	Checksum error in terminal configuration file
307	Passed value to <b>FDEVICECONTROL</b> less than minimum
308	Passed value to <b>FDEVICECONTROL</b> greater than maximum
309	Passed value to <b>FDEVICECONTROL</b> is unsupported
310	Count to <b>FDEVICECONTROL</b> insufficient for return information
311	Count to <b>FDEVICECONTROL</b> greater than available store information
312	Passed special character has previously defined function

## Operation Notes

FCHECK is used to determine the error conditions of the last failed FOPEN intrinsic call (even if a file number was not returned) by setting the *filenum* parameter to zero. In this case, only *fserrorcode* returns valid information.

Do not use FCHECK to determine error conditions of a last failed HPFOPEN call; error conditions are returned in the HPFOPEN *status* parameter.

## Condition Codes

- CCE (2) Request granted.
- CCG (0) Not returned.
- CCL (1) Request denied. The file number passed by *filenum* is invalid, or a bounds violation occurred while processing this request (*fserrorcode*=73).

## FCHECK

### Related Information

Intrinsics      None

Commands      None

Manuals      *Accessing Files Programmer's Guide* (32650-90017), *Using KSAM XL* (32650-90168), and *MPE XL Asynchronous Serial Communications Programmer's Reference Manual* (32022-90012)



---

## FCLOSE

NM and CM callable.

Terminates access to a file on any device.

### Syntax

I16V      I16V      I16V FCLOSE( <i>filenum</i> , <i>disposition</i> , <i>securitycode</i> );
--

### Parameters

- filenum*      **16-bit signed integer by value (required)**  
 Passes the file number of the file to be closed.
- disposition*      **16-bit signed integer by value (required)**  
 Passes the disposition of the file, significant only for files on disk and magnetic tape; ignored for spooled and *hot* printers.  
 (ASC) This parameter is not meaningful for asynchronous devices.

---

**Note**      This *disposition* can be overridden by a corresponding parameter in a FILE command entered prior to program execution.

---

The *disposition* options are:

Bits	Value/Meaning
13:3	Domain disposition:
000	No change. The <i>disposition</i> remains as it was before the file was opened. If the file is new, it is deleted by FCLOSE; otherwise, the file is assigned to the domain it belonged to previously. An unlabeled tape file is rewound and a labeled tape is rewound and unloaded.
001	Close as a permanent file. If the file is a disk file, it is saved in the system file domain. A new or old temporary file on disk has an entry created for it in the system file directory. If a file of the same name already exists in the directory, an error code is returned and the file remains open. If the file is a permanent file on disk, this domain disposition has no effect. If the file is stored on magnetic tape, the tape is rewound and unloaded.

## FCLOSE

Bits	Value/Meaning
13:3	Domain disposition (Continued):
010	Close as a temporary job file (rewound). The file is retained in your temporary (job/session) file domain and can be requested by any process within your job/session. If the file is a disk file, the file name is checked. If a file of the same name already exists in the temporary file domain, an error code is returned and the file remains open. If a file resides on unlabeled magnetic tape, the tape is rewound. If the file resides on labeled magnetic tape, the tape is backspaced to the beginning of the opened file.
011	Close as a temporary job file (not rewind). This option has the same effect as domain disposition 010, except that tape files are not rewound. In the case of unlabeled magnetic tape, if this <b>FCLOSE</b> is the last done on the device (with no other <b>FOPEN/HPFOPEN</b> calls outstanding), the tape is rewound and unloaded. If the file resides on a labeled magnetic tape, the tape is positioned to the beginning of the next file on the tape.
100	Release the file. The file is deleted from the system.
101	Makes a permanent standard disk file temporary (valid only for standard disk files with either fixed-length, variable-length, or undefined-length record formats). The file is removed from the permanent file directory and inserted into the temporary file directory. (PM capability is required for this option.)

Bits	Value/Meaning
11:2	Disk space disposition (valid only for standard disk files with either fixed-length, undefined-length, or variable-length record formats): <ul style="list-style-type: none"> <li>00 Does not return any disk space allocated beyond the end-of-file marker.</li> <li>01 Returns any disk space allocated beyond the end-of-file (EOF) marker to the system. The EOF becomes the file limit; records cannot be added to the file beyond the EOF.</li> <li>10 Returns any disk space allocated beyond the end-of-file (EOF) marker to the system. The file limit remains the same; records can be added to the file beyond EOF, up to the file limit. The disk space disposition takes effect on each FCLOSE.</li> </ul>
0:11	Reserved for the system.

**Note**


---

When a file is opened with the FOPEN/HPFOPEN intrinsic, the file count (maintained by the operating system for each file) is incremented by one; when a file is closed, the file count is decremented by one. If more than one FOPEN/HPFOPEN is in effect for a particular file, its domain disposition is saved but not affected by the FCLOSE call until the file count is decremented to zero; the saved *disposition* is the smallest nonzero *disposition* parameter specified for all FCLOSE calls issued against the file.

For example, the file XYZ is opened three successive times by a process. The first FCLOSE *disposition* is 1, the second is 2, and third is 3. On the last FCLOSE, a *disposition* of 1 will occur.

---

## FCLOSE

*securitycode* 16-bit signed integer by value (required)

Returns the type of security initially applied to the file (significant for new permanent files only). The valid options are:

Value	Meaning
0	Unrestricted access; can be accessed by any user, unless prohibited
1	Private file creator security; can be accessed only by the creator

The *securitycode* parameter selects the security mask assigned to new permanent files whether or not the file is within an MPE group.

(ASC) This parameter is not meaningful for asynchronous devices.

### Operation Notes

FCLOSE deletes buffers and control blocks where the process accessed the file. It also deallocates the device where the file resides, and it can change the *disposition* of the file. If FCLOSE calls are not issued for all files opened by the process, the calls are issued automatically by the operating system when the process terminates. All magnetic tape files are left offline after FCLOSE calls, to indicate that the system operator can remove them.

For circular files, deletion of disk space beyond the end of file is not allowed. Such a request is not an error but no operation is performed. For RIO files, cutback disposition works the same as for flat files: if you have any access to the file other than read and you call FCLOSE with `disp=%10`, the file is closed, the space beyond the EOF is returned to the system, the file's limit is changed to EOF, and no error is returned.

The FCLOSE intrinsic can be used to maintain position when creating or reading a labeled tape file that is part of a volume set:

- If the file is closed with a *disposition* code of 3, the tape does not rewind but remains positioned at the next file.
- If the file is closed with a *disposition* code of 2, the tape rewinds to the beginning of the file, but is not unloaded.
- A subsequent request to open the file does not reposition if the sequence (*seq*) subparameter of *formmsg* in FOPEN/HPFOPEN specifies NEXT or the default (1).
- A *disposition* code of 1 (rewind and unload) implies the close of an entire volume set.
- If unlabeled magnetic tape is closed with a *disposition* code of 0, 1, or 4, and the tape was written to while open, FCLOSE writes three EOFs at the end of the tape before performing a rewind or rewind/unload. This ensures that all tapes have an acceptable number of EOF marks at the end. The three EOFs are written only after the last FCLOSE occurs, before the rewind, and only if the tape was written on.
- For circular files, deletion of disk space beyond the end-of-file is not allowed.
- If the new file status time stamp has changed, FCLOSE updates this time stamp in the file label on disk.

(ASC) For serial devices:

- If one of the following device settings is altered programmatically, the device returns to the system default at **FCLOSE**:
  - Disable read timeout (value set by **FCONTROL** *itemnum*=4).
  - Disable subsystem break (enabled through **FCONTROL** *itemnum*=17).
  - Enable printing a string of three exclamation points (!!!) when a line is deleted (disabled through **FCONTROL** *itemnum*=35).
  - Disable transparent editing (enabled through **FCONTROL** *itemnum*=41).

These are only the control actions taken by the device control software; additional changes can result from actions taken by other software modules. For example, if your program disables the system break feature, break is reenabled on **FCLOSE** by the CI software.

- For most device settings that can be programmatically altered, ensure that the altered characteristics are returned to their original settings when the program ends unless the altered setting define the way the device should act. For example, if the speed setting of a device is programmatically altered, you should return the terminal to its original setting unless the new speed setting should remain in effect for other files opened against the device.
- When all files have been closed on a device and the device is no longer under the control of a program or session, all device characteristics are returned to those specified by the device's configuration.

Directories exist only in the permanent file domain. Permanent (001) and no change are the only values supported for **disposition** (13:3) bits when **filenum** references a directory. Attempts to move a directory from the permanent file domain to either the new or temporary file domain will fail with a CCL condition code and the directory remains open. Directories cannot be deleted by **FCLOSE**. Attempts to release a directory will fail with a CCL condition and the directory remains open.

## Condition Codes

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. The file was not closed; an incorrect <i>filenum</i> was specified or another file with the same name and <i>disposition</i> exists. Any outstanding write I/Os that failed (for example, buffered writes that are done in background) or an illegal <i>disposition</i> .

## Related Information

Intrinsics	<b>FOPEN</b>
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017) and <i>MPE XL Asynchronous Serial Communications Programmer's Reference Manual</i> (32022-90012)

---

## 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.

### Syntax

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

### Parameters

*filenum*      **16-bit signed integer by value (required)**

Passes the file number of the file for which the control operation is to be performed.

*itemnum*      **16-bit signed integer by value (required)**

Specifies which operation is to be performed (refer to Table 4-3).

*item*          **type varies (required)**

Passes/returns a value associated with a control operation as indicated by the corresponding *itemnum* parameter (refer to Table 4-3).

(KSAM) This parameter is ignored, but must be specified to satisfy internal requirements.

Table 4-3. FCONTROL Itemnum/Item Values

Itemnum	Mnemonic	Item Description
0	U16	<p>General device control:</p> <p>The value specified is passed to the appropriate device driver. A value from the driver is returned in <i>item</i>. Not valid for spooled device files.</p>
1	U16	<p>Carriage control (CCTL):</p> <p>Passes a carriage control directive to the driver of the specified nonspooled device file, regardless of whether the file is opened CCTL or NOCCTL. For spooled devices, use <b>FWRITE</b> to pass carriage control directives. (Refer to the <b>FWRITE</b> intrinsic for these directives.)</p> <p>If <i>item</i> passes either the prespacing (%101 or %401) or postspacing (%100 or %400) carriage control directives, one of the following values is returned in <i>item</i>, indicating the setting of the device prior to the <b>FCONTROL</b> call:</p> <p>0    Postspacing 1    Prespacing</p> <p>If both a mode control directive and a carriage control directive need to be specified, an additional call to <b>FWRITE</b> or <b>FCONTROL</b> must be made.</p> <p>This <i>itemnum</i> is ignored for byte stream files. It is not applicable to directories since applications are not permitted to write to directory files.</p> <p>Default: postspacing with automatic page eject (except for FORTRAN77/XL and COBOLII/XL: prespacing with automatic page eject).</p>
2	I16	<p>Complete I/O:</p> <p>Ensures that requested I/O has been physically completed. Valid only for buffered files. Posts the block being written (full or not). Ignored for message files.</p> <p><i>Item</i> is ignored.</p> <p>This <i>itemnum</i> is supported for byte stream files. It is not applicable to directories since applications are not permitted to write to directory files.</p> <p>(KSAM) A checkpoint record is written. In the event of a system crash, recovery would be done to this state of the files.</p>
3	U16	<p>Device status:</p> <p>Returns a record containing information about the state of the device associated with the file immediately after the last I/O operation (including <b>HPFOPEN/FOPEN</b>) on the file. The record size and contents are device-dependent.</p>

**FCONTROL**

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
4	U16	<p>Set read timeout interval:</p> <p>Passes the timeout interval, in seconds, to be applied to the next read from the specified file. The maximum value allowed is 65,535 seconds. If input is requested from a file but is not received in this interval, the <b>FREAD</b> request terminates with CCL (1). The interval is specified in seconds and returned no value. If this interval is zero, any previously established interval is cancelled, and no timeout occurs.</p> <p>If a timeout value is being set for a read posted against a device connected through a PAD, add enough time to compensate for time delays that may be caused by the network.</p> <p>A timeout value should be used for programs reading from an unattended device to prevent “hangs”. Timeouts can be used to terminate binary reads (only as a safeguard to prevent a program from waiting forever for a read to complete).</p> <p>Only valid for terminal and message files. Only affects the next read if the file is being read from the terminal; it must be reissued for each read. If this code is applied to a message file, <i>item</i> specifies the length of time that a process waits when reading from an empty file or writing to a full one and the timeout remains enabled until it is explicitly cancelled.</p> <p>During block mode reads, the timer halts when a DC2 character is received. At this time, the block mode read timer is activated by the system; its values are not user changeable.</p>
5	U16	<p>Reposition file at its beginning:</p> <p>The next record, read or written, is the first record in the file. Not valid for files accessed as append-only. On a labeled magnetic tape file or serial disk file, the tape is positioned at the beginning of the opened file, and not necessarily at the beginning of the volume. Rejected for spooled <b>DATA</b> tapes.</p> <p><i>Item</i> is ignored.</p> <p>This <i>itemnum</i> is supported for byte stream files and all types of directories.</p> <p>(KSAM) The file is repositioned to the first logical record; the record with the lowest value in the current key.</p>



Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
6	U16	<p>Write end-of-file:</p> <p>Marks the end-of-file (EOF) on disk or magnetic tape and is effective only for those devices:</p> <ul style="list-style-type: none"> <li>■ If used for a disk file it writes a logical end-of-data marker where the file was last accessed. The disk file label is updated and written to disk.</li> <li>■ If used for a file residing on unlabeled magnetic tape, a tape mark is written at that current position.</li> <li>■ If used for message files, it verifies the state of the file by writing the file label and buffer area to disk; ensures the message file can survive system crashes. No EOF is written.</li> <li>■ If used for KSAM files, it performs the function of <i>itemnum</i>=2 and writes the file label. This guarantees that the end-of-file is correct and the extent bit map is updated.</li> </ul> <p>Rejected for spooled <b>DATA</b> tapes and labeled magnetic tape files.</p> <p><i>Item</i> is ignored.</p> <p>This <i>itemnum</i> is supported for byte stream files. It is not applicable to directories since applications are not permitted to write to directory files.</p>
7	U16	<p>Space forward to tape mark:</p> <p>Moves a magnetic tape forward until a tape mark is encountered. If used on labeled magnetic tapes, the tape is positioned at the beginning of user trailer labels, if any. Rejected for spooled <b>DATA</b> tapes.</p> <p><i>Item</i> is ignored.</p> <p>(KSAM) Not used for KSAM XL files. For KSAM/3000 files, it clears the key and data buffers of all information and reads the first 2 sectors of the key file from disk to buffer.</p>
8	U16	<p>Space backward to tape mark:</p> <p>On unlabeled tapes, moves a magnetic tape backward until a tape mark is encountered. If used on labeled tapes, the tape is positioned at the beginning of user header labels, if any. Rejected for spooled <b>DATA</b> tapes.</p> <p><i>Item</i> is ignored.</p> <p>(KSAM) Not used for KSAM files.</p>
9	U16	<p>Rewind and unload tape:</p> <p>Repositions a magnetic tape file at its beginning and places the tape offline. Not allowed for labeled tapes. Rejected for spooled <b>DATA</b> tapes and labeled magnetic tape files.</p> <p><i>Item</i> is ignored.</p>

**FCONTROL****Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
10	U16	<p>Change line speed of a device:</p> <p>Passes a value specifying a new line speed for both input and output.</p> <p>The value passed must be expressed in characters per second (1 character per second = 10-bits per second). The supported choices are:</p> <p>30 (300 bps)  120 (1200 bps)  240 (2400 bps)  960 (9600 bps)  1920 (19200 bps)</p> <p>Any setting other than those listed returns CCL (1).</p> <p>The speed associated with the device must match the physical speed settings of the device/modem. If the speed setting is changed programmatically, a request that the speed setting of the device be manually changed should be made.</p> <p>If issued against a terminal connected through PAD, CCE (2) is returned, but there is no effect on the transmission speed, and a meaningless value is returned in <i>item</i>.</p> <p>When a device is initially opened programmatically, the speed is set to the default device speed. If the device is opened as a logon device (configured with speed and parity sensing enabled), the speed setting is sensed from the first carriage return character received. The default device speed is configured through the Node Management Configuration Manager (NMMGR) or the OpenView DTC Manager (networks managed by an OpenView workstation). Reconfigure the device to permanently change the speed.</p> <p>Not reset in break mode; not reset after <b>FCLOSE</b>.</p> <p><i>Item</i> is ignored.</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
11	U16	<p>Change line speed of a device:</p> <p>Passes a value specifying a new line speed for both input and output.</p> <p>The value passed must be expressed in characters per second (1 character per second = 10-bits per second). The supported choices are:</p> <p>30 (300 bps)  120 (1200 bps)  240 (2400 bps)  960 (9600 bps)  1920 (19200 bps)</p> <p>Any setting other than those listed returns CCL (1).</p> <p>The speed associated with the device must match the physical speed settings of the device/modem. If the speed setting is changed programmatically, a request that the speed setting of the device be manually changed should be made.</p> <p>If issued against a terminal connected through PAD, CCE (2) is returned, but there is no effect on the transmission speed, and a meaningless value is returned in <i>item</i>.</p> <p>When a device is initially opened programmatically, the speed is set to the default device speed. If the device is opened as a logon device (configured with speed and parity sensing enabled), the speed setting is sensed from the first carriage return character received. The default device speed is configured through the Node Management Configuration Manager (NMMGR) or the OpenView DTC Manager (networks managed by an OpenView workstation). Reconfigure the device to permanently change the speed.</p> <p>Not reset in break mode; not reset after FCLOSE.</p> <p><i>Item</i> is ignored.</p>

**FCONTROL****Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
12	U16	<p>Enable echo facility:</p> <p>Returns a value indicating the previous echo status:</p> <p>0    Echo ON 1    Echo OFF</p> <p>When enabled, all characters transmitted to the DTC are echoed onto the terminal's screen (DTC echo). When the system is initialized, all characters are echoed except for XON, XOFF, NULL, DEL, and DC2. During binary reads, all characters are passed as data and echoed. For this reason, echo should be disabled while in binary mode.</p> <p>During block mode processing, echo should be disabled and local echo should be used to cause the terminal to write data back to the screen. If VPLUS is being used for block mode applications, this is done automatically. For all other types of block mode, disable echo before the block mode read begins.</p> <p>Echo should be disabled if a terminal user is entering data that should not appear on the screen (for example, passwords or lockwords).</p> <p>Echo should be disabled when connecting some nonsupported devices to an asynchronous port.</p> <p>Reset in break mode; not reset after <b>FCLOSE</b>. (Default)</p>
13	U16	<p>Disable echo facility:</p> <p>Returns a value indicating the previous echo status:</p> <p>0    Echo ON 1    Echo OFF</p> <p>When enabled, all characters transmitted to the DTC are echoed onto the terminal's screen (DTC echo). When the system is initialized, all characters are echoed except for XON, XOFF, NULL, DEL, and DC2. During binary reads, all characters are passed as data and echoed. For this reason, echo should be disabled while in binary mode.</p> <p>During block mode processing, echo should be disabled and local echo should be used to cause the terminal to write data back to the screen. If VPLUS is being used for block mode applications, this is done automatically. For all other types of block mode, disable echo before the block mode read begins.</p> <p>Echo should be disabled if a terminal user is entering data that should not appear on the screen (for example, passwords or lockwords).</p> <p>Echo should be disabled when connecting some nonsupported devices to an asynchronous port.</p> <p>Reset in break mode; not reset after <b>FCLOSE</b>.</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
14	U16	<p>Disable system break function:</p> <p>Enter a value of 1 to disable the system break function.</p> <p>System break is enabled by default for any terminal on which a session is active. If a user presses <b>(BREAK)</b> or a process calls the <b>CAUSEBREAK</b> intrinsic, the system attempts to interrupt the processing and place the terminal at the command interpreter level. Many system commands and program commands (that invoke subsystems or run user programs) are breakable.</p> <p>When the system break is enabled and received, EOR, read timer, terminal mode, and echo values are saved by the DTS software. This allows restoration when normal processing is resumed. If a system break is entered during processing of a character mode read or write, data is lost. Any read that is interrupted by a break is reissued by the system after a <b>RESUME</b> command is entered.</p> <p>Some application programs change the settings of terminals and/or the characteristics of their device files. In these cases, it may be undesirable to allow system break processing to occur. For example, disable system break when using block mode. This prevents data loss or corruption that could occur if <b>(BREAK)</b> is pressed during a block mode read.</p> <p>Reset after <b>FCLOSE</b>. If <b>(BREAK)</b> is pressed while system break is disabled, no action is taken by any level of software. System break has no effect on a device with no active session.</p>
15	U16	<p>Enable system break function:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>System break is enabled by default for any terminal on which a session is active. If a user presses <b>(BREAK)</b> or the <b>CAUSEBREAK</b> intrinsic is called, the system attempts to interrupt the processing and place the terminal at the command interpreter level. Many system commands and program commands (that invoke subsystems or run user programs) are breakable.</p> <p>When the system break is enabled and received, EOR, read timer, terminal mode, and echo values are saved by the DTS software. This allows restoration when normal processing is resumed. If a system break is entered during processing of a character mode read or write, data is lost. Any read that is interrupted by a break is reissued by the system after a <b>RESUME</b> command is entered.</p> <p>Some application programs change the settings of terminals and/or the characteristics of their device files. In these cases, it may be undesirable to allow system break processing to occur. For example, disable system break when using block mode. This prevents data loss or corruption that could occur if <b>(BREAK)</b> is pressed during a block mode read.</p> <p>Reset after <b>FCLOSE</b>. (Default) If <b>(BREAK)</b> is pressed while system break is disabled, no action is taken by any level of software. System break has no effect on a device with no active session.</p>

**FCONTROL**

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
16	U16	<p>Disable subsystem break function:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>Subsystem break is initially disabled on all devices; it must be enabled before it can be used. It is available only for devices running sessions and has no effect on programmatically controlled devices. Subsystem break is reset to disabled at <b>FCLOSE</b>.</p> <p>When enabled, the user can stop a program-local or subsystem-local command. It is invoked by the subsystem break character. (The default subsystem break character is <b>CTRL Y</b>.)</p> <p>When enabled for a terminal connected through PAD, the subsystem break must be followed by <b>RETURN</b> to signal the PAD that data is ready.</p> <p>Before enabling the subsystem break, call the <b>XCONTRAP</b> intrinsic to arm the subsystem break trap procedure. When a subsystem break is entered, the subsystem break trap is disarmed. The <b>RESETCONTROL</b> intrinsic must be called to allow another subsystem break to occur.</p> <p>Subsystem break processing is summarized as follows:</p> <ol style="list-style-type: none"> <li>1. A user-written procedure must be provided defining how the subsystem break trap handler will react upon receipt of a subsystem break. The user program must contain: <ul style="list-style-type: none"> <li>■ The user written procedure.</li> <li>■ A call to <b>XCONTRAP</b>, specifying the external label of the procedure; this arms the subsystem break trap.</li> <li>■ A call to <b>FCONTROL itemnum=17</b>, enabling the subsystem break function.</li> <li>■ A call to <b>RESETCONTROL</b> when the program is ready to receive another subsystem break.</li> </ul> </li> <li>2. A new subsystem break character can be defined through <b>FCONTROL itemnum=41</b> if the terminal is placed in transparent mode or through <b>FDEVICECONTROL</b>, which allows a subsystem break character to be defined without placing the device in the transparent mode.</li> <li>3. At any time, <b>FCONTROL</b> can be called to disable the subsystem break.</li> </ol> <p>Reset in break mode; reset after <b>FCLOSE</b>. (Default)</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
17	U16	<p>Enable subsystem break function:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>Subsystem break is initially disabled on all devices; it must be enabled before it can be used. It is available only for devices running sessions, and has no effect on programmatically controlled devices. Subsystem break is reset to disabled at <b>FCLOSE</b>.</p> <p>When enabled, the user can stop a program-local or subsystem-local command. It is invoked by the subsystem break character. (The default subsystem break character is <b>(CTRL)Y</b>.)</p> <p>When enabled for a terminal connected through PAD, the subsystem break must be followed by <b>(RETURN)</b> to signal the PAD that data is ready.</p> <p>Before enabling the subsystem break, call the <b>XCONTRAP</b> intrinsic to arm the subsystem break trap procedure. When a subsystem break is entered, the subsystem break trap is disarmed. The <b>RESETCONTROL</b> intrinsic must be called to allow another subsystem break to occur.</p> <p>Subsystem break processing is summarized as follows:</p> <ol style="list-style-type: none"> <li>1. A user-written procedure must be provided defining how the subsystem break trap handler will react upon receipt of a subsystem break. The user program must contain: <ul style="list-style-type: none"> <li>■ The user written procedure.</li> <li>■ A call to <b>XCONTRAP</b>, specifying the external label of the procedure; this arms the subsystem break trap.</li> <li>■ A call to <b>FCONTROL itemnum=17</b>, enabling the subsystem break function.</li> <li>■ A call to <b>RESETCONTROL</b> when the program is ready to receive another subsystem break.</li> </ul> </li> <li>2. A new subsystem break character can be defined through <b>FCONTROL itemnum=41</b> if the terminal is placed in transparent mode or through <b>FDEVICECONTROL</b>, which allows a subsystem break character to be defined without placing the device in the transparent mode.</li> <li>3. At any time, <b>FCONTROL</b> can be called to disable the subsystem break.</li> </ol> <p>Reset in break mode; reset after <b>FCLOSE</b>. (Default)</p>

## FCONTROL

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
18	U16	Disable tape option (Default) <i>Item</i> is ignored.
19	U16	Enable tape option: <i>Item</i> is ignored.
20	U16	Disable the terminal input timer: Reset in break mode; reset after <b>FCLOSE</b> . (Default) <i>Item</i> is ignored.
21	U16	Enable the terminal input timer: Reset in break mode; reset after <b>FCLOSE</b> . <i>Item</i> is ignored.
22	U16	Read the terminal input timer: <i>Item</i> returns the measured time duration of the last read.  The result of the read duration timer is returned in hundredths of a second, up to 655.35 seconds.  The operating system times each and every read, therefore, this call should immediately follow the read of interest with no subsequent reads between.
23	U16	Disable parity checking and generation: <i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.  If a call is issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not take place.  Parity can be used for terminals opened programmatically and for those in session mode. The default for these is to operate in 8-bit pass-through mode (no parity).  If a terminal detects a parity error, it displays the PAD character.  If a parity error is detected by ASC software, the current read is completed in error, and no read data is returned. The ASC software reports the error to the program reading the data.



Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
24	U16	<p>Enable parity checking and generation</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>If a call is issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not take place.</p> <p>Parity can be used for terminals opened programmatically and for those in session mode. The default for these is to operate in 8-bit pass-through mode.</p> <p>If a terminal detects a parity error, it displays the PAD character.</p> <p>If a parity error is detected by ASC software, the current read is completed in error, and no read data is returned. The ASC software reports the error to the program reading the data.</p> <p>When enabled, the ASC software generates parity on outgoing data and checks the parity errors on incoming data. After the parity is checked, the parity bit is always set to zero. (The program using the data does not need parity data.) Parity checking is handled the same in block mode and in character mode.</p> <p>When enabling parity, ensure that the program also requests the terminal operator to change the physical parity setting of the terminal to the new setting of the ASC software. Also, if the terminal is to check parity on incoming data, the local terminal control (used to turn parity checking on or off) should be set to ON.</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description																								
25	U16	<p>Define additional end-of-record characters:</p> <p><i>Item</i> passes the value of a character to be used as an additional end-of-record (AEOR) character. The octal or decimal code for the character must be in the right byte of the parameter; the left byte is ignored. If the NULL character (%0) is specified, default EOR conditions are restored.</p> <p>Normally, character mode reads using standard editing are terminated by a <b>RETURN</b>. The system echoes the carriage return and sends a linefeed to the terminal, the cursor is positioned at the beginning of the next line, ready for the next read. If an AEOR is assigned, an EOR character still terminates a read as usual.</p> <p>When a read is terminated by an AEOR, the AEOR character is included in both the data and byte counts. No carriage return or line feed is sent to the terminal. The read terminates with an error condition indicating that the AEOR character was encountered.</p> <p>The AEOR character is also recognized in transparent editing mode, along with a user-defined EOR character. (This replaces the normal EOR character.)</p> <p>To disable the AEOR character, indicate <i>item</i>=0.</p> <p>The following are not recognized as AEOR characters:</p> <table border="0"> <thead> <tr> <th>ASCII Characters</th> <th>Octal Code</th> <th>Hex Code</th> <th>Terminal Keys</th> </tr> </thead> <tbody> <tr> <td>NUL (null)</td> <td>%0</td> <td>\$0</td> <td><b>CTRL</b>@</td> </tr> <tr> <td>DC1 (XON)</td> <td>%21</td> <td>\$11</td> <td><b>CTRL</b>Q</td> </tr> <tr> <td>DC3 (XOFF)</td> <td>%23</td> <td>\$13</td> <td><b>CTRL</b>S</td> </tr> <tr> <td>DEL (Rubout)</td> <td>%31</td> <td>\$19</td> <td><b>DEL</b></td> </tr> <tr> <td>Subsystem Break</td> <td>%177</td> <td>\$77F</td> <td><b>CTRL</b>Y (default)</td> </tr> </tbody> </table> <p>Note: Avoid using characters that have a special meaning specifically, carriage return, linefeed, cancel, backspace, DC2, or escape. These characters will be recognized and processed, but they can result in unpredictable and unsuccessful results.</p> <p>Do not use <b>CTRL</b>A as an AEOR character at the console.</p>	ASCII Characters	Octal Code	Hex Code	Terminal Keys	NUL (null)	%0	\$0	<b>CTRL</b> @	DC1 (XON)	%21	\$11	<b>CTRL</b> Q	DC3 (XOFF)	%23	\$13	<b>CTRL</b> S	DEL (Rubout)	%31	\$19	<b>DEL</b>	Subsystem Break	%177	\$77F	<b>CTRL</b> Y (default)
ASCII Characters	Octal Code	Hex Code	Terminal Keys																							
NUL (null)	%0	\$0	<b>CTRL</b> @																							
DC1 (XON)	%21	\$11	<b>CTRL</b> Q																							
DC3 (XOFF)	%23	\$13	<b>CTRL</b> S																							
DEL (Rubout)	%31	\$19	<b>DEL</b>																							
Subsystem Break	%177	\$77F	<b>CTRL</b> Y (default)																							

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
26	U16	<p>Disable binary mode:</p> <p><i>Item</i> has no meaning, enter a value of 0 to satisfy internal requirements.</p> <p>Ignored for terminals connected through PAD.</p> <p>Binary mode is disabled by default.</p>
27	U16	<p>Enable binary mode:</p> <p><i>Item</i> has no meaning, enter a value of 0 to satisfy internal requirements.</p> <p>If this call is issued to a device connected through PAD, it is ignored; however, the next <b>FREAD</b> posted to the device returns an error.</p> <p>When binary mode is enabled, no special characters are processed; all characters are considered data and are passed through without any terminal control actions being taken. After a binary read, no carriage return or line feed is sent to the terminal. All carriage control directives are ignored on writes. Unless echo is disabled, any ASCII DC3 (XOFF) character passed in data is echoed to the terminal, and the terminal is suspended. Block mode transfers cannot be made.</p> <p>Binary mode does not take effect until a read is posted to the device following the <b>FCONTROL</b> call. If binary mode is enabled for both reads and writes, post a 0 byte read immediately after the call to ensure binary mode is enabled.</p> <p>Binary mode is useful for transferring 8-bit data to and from a terminal. Because all 8-bits are considered to be data, binary mode is not compatible with parity checking; parity must be disabled.</p> <p>In session mode, a system break restores standard editing at the terminal. If <b>RESUME</b> is entered following the system break, binary mode is restored when the first read is posted.</p> <p>Binary reads are terminated on byte count, as specified in the <b>FREAD</b>, <b>READ</b>, or <b>READX</b> call. Errors and read timeouts also terminate binary reads, but an error code is returned and no data is transferred.</p>

**FCONTROL**

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
28	U16	<p>Disable user block mode:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>Ignored for terminals connected through PAD.</p> <p>User block mode is not supported with terminal type 18.</p> <p>Under system default conditions, block mode processing occurs through a method called Hewlett-Packard block mode. The system controls the block mode handshake if block mode transfers are being made and the program does not need to worry about the data transfer protocol. The host transmits a DC1 character to inform the terminal that the system is ready to receive data. The terminal responds with a DC2 character to inform the system that the next data transfer is a block mode transmission. The terminal then waits for the host to send a second DC1 character to initiate the read.</p> <p>When transmitting data in block mode, the terminal must be configured to be compatible with the system and the application requesting the block mode transfer.</p> <p>Note: Data overruns can occur during block mode transfers. The program must check for successful completion of each read, and retry as required. Use of timers during block mode is encouraged to eliminate port “hangs” due to a data overrun occurring when the last character is read. The standard block mode read timer is disabled when the user block mode protocol is enabled.</p> <p>Not reset in break mode; not reset after <b>FCLOSE</b>. (Default)</p>
29	U16	<p>Enable user block mode:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>Ignored for terminals connected through PAD.</p> <p>User block mode is not supported with terminal type 18.</p> <p>When user block mode is enabled and block mode transfers are being made, the program can intervene in the block mode handshake. A user block mode transaction begins with the DC1 character and then the program takes control of the handshake until the DC2 character is encountered. The program can perform additional terminal control functions, such as positioning the terminal cursor, performing status requests, or collating additional space for buffers before continuing with the data transfer. When the program is ready to receive block mode data, an additional read must be issued (DC1 character) to complete the handshake.</p> <p>When transmitting data in block mode, the terminal must be configured to be compatible with the system and the application requesting the block mode transfer.</p> <p>Note: Data overruns can occur during block mode transfers. The program must check for successful completion of each read, and retry as required. Use of timers during block mode is encouraged to eliminate port “hangs” due to a data overrun occurring when the last character is read. The standard block mode read timer is disabled when the user block mode protocol is enabled.</p> <p>Not reset in break mode; not reset after <b>FCLOSE</b>. Under default conditions, user block mode is disabled.</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
34	U16	<p>Enable line deletion response:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>In character mode, with standard editing, the computer will output !!!, carriage return, and line feed when the line deletion (cancel) character is entered. This is referred to as <b>(CTRL)</b>X echo (the default line deletion character).</p> <p>Not reset in break mode; reset after <b>FCLOSE</b>. (Default)</p> <p>For devices connected through PAD the line deletion response is never printed in response to the line deletion character. If this call is issued, condition code CCE (2) is returned, but no device control action occurs.</p>
35	U16	<p>Disable line deletion response:</p> <p><i>Item</i> has no meaning; enter a value of 0 to satisfy internal requirements.</p> <p>The line deletion response can be suppressed by calling this parameter. The data is deleted in the input buffer, but no !!!, carriage return, line feed is issued.</p> <p>Not reset in break mode; reset after <b>FCLOSE</b> (enables line deletion suppression, the system default condition).</p> <p>For devices connected through PAD, the line deletion response is never printed in response to the line deletion character. If this call is issued, condition code CCE (2) is returned, but no device control action occurs.</p>
36	U16	<p>Set parity:</p> <p><i>Item</i> passes a value representing the specified parity condition and then returns the value of a previous setting:</p> <ul style="list-style-type: none"> <li>0 No parity check on input, eighth bit set to 0 on output (parity type = zeros)</li> <li>1 No parity check on input, eighth bit set to 1 on output (parity type = ones)</li> <li>2 Even parity checked on input, even parity generated on output (parity type = even)</li> <li>3 Odd parity checked on input, odd parity generated on output (parity type = odd)</li> </ul> <p>When specified, this option changes the terminal parity setting, and the previous setting is returned to the call as the new value of <i>item</i>. It is possible to determine a terminal's parity setting with parity disabled.</p> <p>Parity cannot be used with 8-bit character sets.</p> <p>Ignored for devices connected through PAD.</p>

## FCONTROL

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
37	U16	<p>Set terminal type and speed:</p> <p><i>Item</i> passes a value specifying line speed and terminal type. Bits (0:11) specify the line speed; bits (11:5) specify the terminal type.</p> <p>Ensure that the speed associated with a device matches the physical speed setting of the device and modem (if part of the connection). If the device speed is being changed programmatically, request that the speed setting be manually changed on the device.</p> <p>If an attempt is made to set an unsupported speed or terminal type, condition code CCL (1) is returned.</p> <p>Do not use this call to specify a user-defined terminal type.</p> <p>If used for devices connected through PAD, the call changes the terminal type setting, but does not change how the device actually operates. Condition code CCE (2) is returned.</p>
38	U16	<p>Set terminal type</p> <p><i>Item</i> passes the value of the system-defined terminal type to be associated with an asynchronous port.</p> <p>Changes the terminal type associated with a port to another system-defined terminal type. The new terminal type remains in effect until another <b>FCONTROL</b> call is made or the device is closed. If the device is closed, the port resets to its configured terminal type.</p> <p>The terminal type specified must be a currently supported system-defined terminal type. If another value is used, condition code CCL (1) is returned.</p> <p>This call cannot be used to change to a user-defined terminal type.</p> <p>If used for a terminal connected through PAD, there is no effect on the terminal control settings of the device. Condition code CCE (2) is returned.</p>
39	U16	<p>Obtain terminal type information</p> <p><i>Item</i> returns the terminal type associated with the asynchronous port. Use this information to get the original setting before calling <i>itemnum</i>=37 or 38.</p> <p>Note: Do not confuse the current terminal type returned with the default terminal type specified during system configuration. Changing the terminal type during logon or through system intrinsics can cause a difference from the configured setting. To reset the configured terminal type, use <i>itemnum</i>=37 and <i>item</i>=0 or close the device.</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
40	U16	<p>Obtain terminal output speed</p> <p><i>Item</i> returns the current line speed associated with an asynchronous port in characters per second. Use this information to get the original setting before calling <i>itemnum</i>=11 or 37.</p> <p>Note: Do not confuse the current line speed returned with the default line speed specified during system configuration. Changing the line speed through system intrinsics can cause a difference from the configured setting. To reset the configured line speed, use <i>itemnum</i>=37 and <i>item</i>=0 or close the device.</p> <p>If used for a terminal connected through PAD, the value returned in <i>item</i> is meaningless. Condition code CCE (2) is returned.</p>
41	U16	<p>Enable transparent editing:</p> <p><i>Item</i> passes a value to be used as a subsystem break character in bits (0:8) and a value to be used as a read terminator in bits (8:8). Returns the values previously assigned for use as subsystem break and read terminator characters.</p> <p>When enabled, allows most special characters to be read and treated as data and a small subset to retain their meaning:</p> <p>AEOR            The AEOR character defined through a previous call to <i>itemnum</i>=25. The read terminates in error, and a call to <b>FCHECK</b> returns an error code of 31. The AEOR character is passed with the data and included in the byte count.</p> <p><b>(BREAK)</b>       Causes a system break (when enabled) and transfers control to the operating system. If the <b>RESUME</b> command is entered, transparent editing is resumed.</p> <p><b>(CTRL)Q</b> (DC1 or XON)       The resume output character of the XON/XOFF handshake. This character is stripped from the input data when XON/XOFF protocol is enabled.</p> <p>DC2            When input as the first character, it is stripped out. A DC1 (<b>(CTRL)Q</b>) is written to the terminal. If not input as the first character, DC2 is a data character.</p> <p><b>(CTRL)S</b> (DC3 or XOFF)       The stop output character of the XON/XOFF handshake. This character is stripped from the input data when XON/XOFF protocol is enabled.</p> <p><i>item</i> bits (0:8)       Specifies the character that replaces the subsystem break character. Any character not defined as a special character in transparent mode (including the standard subsystem break character) can be used. If set to 0, subsystem break is disabled.</p> <p><i>item</i> bits (8:8)       Specifies the character that replaces the line termination character (EOR). Any character not defined as a special character in transparent mode can be used.</p> <p><i>Item</i>=0 disables transparent editing mode and returns to standard editing mode.</p>

**FCONTROL**

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
41 Cont'd		<p>Transparent mode reads can be terminated in any of the following ways:</p> <ul style="list-style-type: none"> <li>■ The read encountered the EOR character defined in <i>item</i>. This character functions like a <b>(RETURN)</b> character in normal reads and is stripped when found. No carriage return or line feed is sent to the terminal.</li> <li>■ The read encountered an AEOR character, defined through a previous call to <b>FCONTROL</b> <i>itemnum</i>=25. The read completes with an error condition and the AEOR character is passed with the input data and included in the byte count. No carriage return or line feed is sent to the terminal.</li> <li>■ The expected byte count is reached. The read is terminated normally, and carriage return and line feed are sent to the terminal.</li> <li>■ The read limit timer expired. All data is lost and the read terminates in error.</li> </ul> <p>Break or subsystem break processing, if not disabled, occurs in transparent editing. During a break, the terminal operates in standard editing mode. The terminal returns to transparent editing mode if the <b>RESUME</b> command is entered. During a subsystem break, the terminal operates the way the application program describes.</p> <p>Parity processing is the same in transparent editing as it is in standard editing.</p> <p>Any <b>FCLOSE</b> call issued against the terminal restores standard editing.</p> <p>If used for a device connected through PAD, data is not forwarded from the terminal immediately after the characters defined as the subsystem break or EOR character are typed. The terminal user has to press <b>(RETURN)</b> to cause data to be transmitted, or wait for the data forwarding timer to expire (if set). The host software recognizes and acts on the characters defined in <i>item</i>.</p>
43	U16	<p>Aborts pending nowait I/O request:</p> <p>For message files, CCG (0) is returned when an outstanding I/O operation has completed. An <b>IOWAIT</b> call must be issued to finish the request.</p>
45	U16	<p>Enable/disable extended wait:</p> <p>Applicable only if <i>filenum</i> specifies a message file. An <i>item</i> value of 1 enables extended wait; permits a reader to wait on an empty file that is not currently opened by any writer, or a writer to wait on a full file that has no reader. This remains in effect until an <b>FCONTROL</b> call is issued with an <i>itemnum</i> 45 and an <i>item</i> value of 0. An <i>item</i> value of 0 specifies that if a second <b>FREAD</b> call is issued and it encounters an empty file that has no reader, it returns an end-of-file condition. Each assessor must call <b>FCONTROL45</b> individually.</p> <p>Default: 0</p>



Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
46	U16	<p>Enable/disable reading writer's ID:</p> <p>Applicable only if <i>filenum</i> specifies a message file.</p> <p>For interprocess communication, an <i>item</i> value of 1 enables reading the writer's ID. Each record read has a two-word header. The first word indicates the type of record:</p> <ul style="list-style-type: none"> <li>0 Data record</li> <li>1 Open record</li> <li>2 Close record</li> </ul> <p>The second word contains the writer's ID number. If the record is a data record, the data follows the header. Open and close records contain no more information. If the <i>item</i> value is 0, reading the writer's ID is disabled. Only data is read to the reader's target area. The open and close records are skipped and deleted by the file system when they come to the head of the message queue, and the two-word header is transparent to the reader.</p> <p>Default: 0</p>
47	U16	<p>Nondestructive read:</p> <p>Applicable only if <i>filenum</i> specifies a message file.</p> <p>If the <i>item</i> value is 1, the next <b>FREAD</b> by this reader does not delete the record. Subsequent <b>FREAD</b> calls are unaffected. When <i>item</i> is specified as 0, the next <b>FREAD</b> deletes the record.</p> <p>Default: 0</p>
48	U32	<p>Arm/disarm software interrupts:</p> <p>Passes a value that arms/disarms the software interrupt mechanism (valid only if <i>filenum</i> specifies a message file). To arm the file's software interrupt, pass the external procedure label (NM plabel) or the interrupt procedure. To disarm the software interrupt, pass a value of zero.</p>

**FCONTROL**

**Table 4-3. FCONTROL Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
51		<p>Append mode:</p> <p>Enables or disables append mode based on the value of the <i>item</i> parameter. Append mode is enabled if <i>item</i> is one (1) and disabled if <i>item</i> is zero (0). Write operations append data at the end of a file while reads and seeks may occur anywhere within the file when POSIX append mode is enabled for that file.</p> <p>Append <i>mode</i> differs from opening a file for append <i>access</i> in several ways:</p> <ol style="list-style-type: none"> <li>1 Append <i>mode</i> can be enabled and disabled while a file is open. Append <i>access</i> is established when opening a file and cannot be changed dynamically for a file number.</li> <li>2 Append <i>access</i> is a process-local attribute affecting only the access of the process open the file. Append <i>mode</i> affects all processes sharing a logical record pointer. (The MULTI and GMULTI options permits logical record pointers to be shared.)</li> <li>3 A file's logical record pointer can be manipulated through FPOINT or FREADDIR when append <i>mode</i> is active. Append <i>access</i> prohibits logical record pointer operations. For example, call to FPOINT and FSPACE result in an access violation for files opened for append <i>access</i>.</li> </ol> <p>This <i>itemnum</i> is supported only for byte stream files. If the file referenced by <i>filenum</i> is not a byte stream file. FCONTROL will fail with a CCL condition code. An error code of 42 (operation inconsistent with device type) is returned by FCHECK when it is called following this FCONTROL error.</p>
52		<p>Set Nonblock</p> <p>This option can be used to set and reset the Nonblock flag of a file. When the Nonblock flag is true, any attempt to perform I/O to the file, results in the I/O procedure returning without blocking. If the I/O can be performed without blocking, then the I/O is completed normally. When Nonblock is false, I/O's will block if necessary. The <i>item</i> parameter must be set to 1 in order to cause the Nonblock flag to be set to true. A value of zero will reset the Nonblock flag. Any other value for the <i>item</i> parameter results in an error being returned.</p> <p>The default value of <i>item</i> is 0.</p> <p>Not all file types support this option.</p>
53		<p>Close File on Exec</p> <p>This option allows the caller to set or reset closure of a specified file when a POSIX <code>exec( )</code> function is called. An <i>item</i> value of 0 causes the file remain open on an <code>exec( )</code>. A value of 1 causes the file to close when an <code>exec( )</code> is performed. All other values of the <i>item</i> parameter return an FSERR 8 - ILLEGAL PARAMETER VALUE error. The default value of <i>item</i> is 0.</p>

Table 4-3. FCONTROL Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
54		<p>This item provides for compatibility with the <code>ioctl( )</code> function found in most UNIX implementations. This option is only valid for streams and network socket files and tape devices. Any attempt to use this option with other file types results in an <code>FSERR 8 - ILLEGAL PARAMETER VALUE</code> error.</p> <p>The <i>item</i> parameter must point to a structure that contains three <code>ioctl( )</code> parameters:</p> <ul style="list-style-type: none"> <li>■ Request</li> <li>■ Arg</li> <li>■ <code>ioctl</code> functional return</li> </ul> <p>The first 32 bits contain a short pointer to the <code>arg</code> parameter. The <code>arg</code> parameter is pointed to by reference due to the variable length depending upon the requested function. The last 32 bits of the structure contains the address of the location to write the functional return. Due to the nature of the functional return, this item is an output only address. In pascal, the structure is defined as follows:</p> <pre> ioctl_parm_ptr = ^ioctl_parm_type;  ioctl_parm_type = record     request      : integer;     arg          : localanyptr;     func_return  : localanyptr;  end;</pre> <p>The caller must correctly fill in the values of the structure before calling the <code>FCONTROL</code> intrinsic. The functions that are supported through this option are defined as standard documentation on the <code>ioctl( )</code> function.</p>

---

**Note**      *Itemnums* 10 through 41 are used in changing terminal characteristics. Included with the definition of the code is an indication, where applicable, of whether the characteristic is reset in break mode or after `FCLOSE`. Characteristics that are reset in break mode are restored when there is a `RESUME` from break mode.

---

### Operation Notes

The `FCONTROL` intrinsic applies to files on disk, tape, terminal, or line printers. There are some special conditions that exist when using `FCONTROL` with files on labeled magnetic tape. Some `FCONTROL` functions cannot be used with labeled tapes, and other functions can produce unexpected results. (Refer to *itemnums* 5, 6, 7, 8, and 9.)

---

**Note**      If you are performing control operations on terminal devices, refer to the *Asynchronous Serial Communications Programmer's Reference Manual* (32022-90002) for details on `FCONTROL` parameters specific to terminal control.

---

## **FCONTROL**

### **Condition Codes**

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred.

### **Related Information**

Intrinsics      FWRITE, FDEVICECONTROL, RESETCONTROL, XCONTRAP

Commands      None

Manuals      *Accessing Files Programmer's Guide* (32650-90017), *Interprocess Communication Programmer's Guide* (32650-90019), *MPE XL Asynchronous Serial Communications Programmer's Reference Manual* (32022-90012), *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364), *Using KSAM XL* (32650-90168), and *Trap Handling Programmer's Guide* (32650-90026)

---

## FDELETE

NM and CM callable.

Deactivates a specified logical record in an RIO file.

### Syntax

I16V    I32V FDELETE( <i>file num</i> , <i>lrec num</i> );
---

### Parameters

- file num*      **16-bit signed integer by value (required)**  
                 Passes the file number of the file to be modified.
- lrec num*      **32-bit signed integer by value (optional)**  
                 Passes the relative logical record to be deactivated.

### Operation Notes

If a record is not specified or the *lrec num* is negative, the next logical record becomes inactive. If the selected record has already been deactivated, CCE (2) is returned. Check to see if an RIO record is inactive by calling the FCHECK intrinsic. An INACTIVE RECORD error indicates that the record selected for the FDELETE was already inactive.

Only applicable for RIO files.

Not used for KSAM files.

### Condition Codes

- CCE (2)      Request granted. No error occurred, but an inactive record may have been encountered.
- CCG (0)      Request denied. An end-of-file (EOF) occurred.
- CCL (1)      Request denied. An access error occurred.

### Related Information

- Intrinsics      None
- Commands      None
- Manuals      *Accessing Files Programmer's Guide* (32650-90017)

---

## 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.

### Syntax

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

### Parameters

*filenum*      **16-bit signed integer by value (required)**

Passes the file number of the device file.

*buffer*        **user-defined structure (required)**

Passes and returns device information:

- Passes information to the device, such as character sets, forms, vertical format control (VFC) information, or device control actions.
- Returns device driver information.

(ASC) The *buffer* is usually a 16-bit integer by reference or, for *controlcode*=192, where *parm*=1, *buffer* is a character array containing the name of the terminal type or printer type file. For *controlcode*=192, *buffer* values are defined in the *parm1* discussion.

*length*        **16-bit signed integer by value (required)**

Passes the length of *buffer*:

- If positive, it indicates the length in half words.
- If negative, it indicates the length in bytes.

(ASC) If *controlcode*=192, *length* is normally set to 1 unless *parm1* is set to 1 (specifying a new terminal or printer type file), then the length of the array is specified in the *length* parameter.

***controlcode*** 16-bit signed integer by value (required)

Passes the item value to be returned. (Refer to Table 4-4.)

(ASC) Passes the number of the operation to be performed. ***Controlcode*** 192 is used to apply control directives to a device file.

Codes 0 through 64 call FCONTROL. Control passes to FCONTROL, and any errors returned will have originated with FCONTROL.

***parm1,***  
***parm2*** 16-bit unsigned integer by value (required)

Passes the value specified in the corresponding ***controlcode***; there are several values for ***parm1*** and ***parm2*** that define the operation in more detail. (Refer to Table 4-4.)

***fserrorcode*** 16-bit unsigned integer by reference (required)

Returns a file system error code number if an error occurs. If no error occurs, ***fserrorcode*** is set to zero. If FDEVICECONTROL detects a bounds violation for ***fserrorcode*** (that is, an address outside your stack area), ***fserrorcode*** is returned unchanged.

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values**

Control-code	Mnemonic	Parm1, Parm2 Description
128	I16V	<p>Character set selection:</p> <p><i>parm1</i> (8:8) Primary character set identification</p> <p><i>parm2</i> (8:8) Secondary character set identification</p> <p>The Hewlett-Packard 268x page printer can contain up to 32 character sets, allowing the use of a variety of fonts, styles, print rotations, and languages. <i>Controlcode</i>=134 is used to download character sets to the printer or to select any two downloaded character sets to be the current primary and secondary character sets.</p> <p>To change to the secondary character set (one character at a time), set the eighth bit of the byte coding for the desired ASCII character. The Hewlett-Packard 268x strips out this bit and prints (in the secondary character set) the character represented by the remaining 7-bit value. To change to the secondary character set for a number of characters and over several lines, insert a <b>CTRL N</b> (shift-in character) in the data. Insert a <b>CTRL O</b> (shift-out character) where the primary character set is to be reactivated.</p>
129	I16V	<p>Logical page activation/deactivation request:</p> <p><i>parm1</i> (0:1) 0 = Ignore the left byte of <i>parm2</i>. 1 = Deactivate the logical page table entry identified in the left byte of <i>parm2</i>.</p> <p><i>parm1</i> (1:1) 0 = Ignore the right byte of <i>parm2</i>. 1 = Activate the logical page table entry identified in the right byte of <i>parm2</i>.</p> <p><i>parm2</i> (0:8) Logical page table entry (0 to 31) to be deactivated. Ignored if <i>parm1</i> (0:1)=0.</p> <p><i>parm2</i> (0:8) Logical page table entry (0 to 31) to be activated. Ignored if <i>parm1</i> (1:1)=0.</p> <p>Logical pages being printed can be canceled or enabled through the activation or deactivation of those pages.</p> <p>Every physical page is composed of one or more logical pages. When the Hewlett-Packard 268x begins to print each physical page, it scans the logical page table (LPT) for the first logical page labeled as <b>ACTIVE</b>. The printer then continues searching the table sequentially for active pages and printing them until it has printed the last active page. At this point, the Hewlett-Packard 268x performs a physical page eject and starts the sequence again. There must be at least one active LPT entry while the Hewlett-Packard 268x is printing.</p>



Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
130	I16V	<p>Relative pen displacement:</p> <p><i>parm1</i>            A 16-bit signed integer containing the desired X-axis displacement, in dots, of the pen from its current position.</p> <p><i>parm2</i>            A 16-bit signed integer containing the desired Y-axis displacement, in dots, of the pen from its current position.</p> <p>No pen movement results from requests to move the pen off the logical page. Because the coordinate system is based upon the current logical page itself and not upon the page's orientation with respect to the printer, do not consider how the page has been rotated when assigning displacement values to <i>parm1</i> and <i>parm2</i>. Since the dot density for the Hewlett-Packard 2680 (180 dots per inch) differs from the Hewlett-Packard 2688 (300 dots per inch), the effects of <i>parm1</i> and <i>parm2</i> are different.</p>
131	I16V	<p>Absolute pen move:</p> <p><i>parm1</i>            An integer containing the X-coordinate, in dots, of the point to move the pen.</p> <p><i>parm2</i>            An integer containing the Y-coordinate, in dots, of the point to move the pen.</p> <p>The values in <i>parm1</i> and <i>parm2</i> are measured from the upper left corner of the logical page. Do not take page rotation into account when assigning coordinates as the printer does not move the pen if the location specified is off the logical page. Since the dot density for the Hewlett-Packard 2680 (180 dots per inch) differs from the Hewlett-Packard 2688 (300 dots per inch), the effects of <i>parm1</i> and <i>parm2</i> are different.</p>
132	I16V	<p>Define job characteristics:</p> <p><i>parm1</i> (0:1)      0 = Ignored                           1 = Printer does not print job separation marks until next open job</p> <p><i>parm1</i> (1:1)      0 = Ignored                           1 = <b>Parm2</b> contains maximum allowable number of copies</p> <p><i>parm2</i>            Significant only if <i>parm1</i> (1:1)=1 (Specifies the maximum number of copies the printer makes, for any one page, for the current job. The default maximum is 32,767.)</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1,Parm2 Description
133	I16V	<p>Define physical page.</p> <p>The following bits are ignored if set to zero:</p> <p><i>parm1</i> (0:1)      1 = Multicopy form overlay feature on  <i>parm1</i> (1:1)      1 = Multicopy form overlay feature off  <i>parm1</i> (2:1)      1 = Reserved for the operating system  <i>parm1</i> (3:1)      1 = Redefine physical page length  <i>parm1</i> (4:1)      1 = Redefine number of copies per page  <i>parm1</i> (5:1)      1 = Reserved for the operating system  <i>parm1</i> (6:1)      1 = Reserved for the operating system  <i>parm1</i> (7:1)      1 = Reserved for the operating system  <i>parm1</i> (8:8)      New physical page length in units of 0.25 inches; cannot be less than 3.0 inches (a value of 12) or greater than 17.0 inches (a value of 68)  <i>parm2</i>              Number of copies to print; if number exceeds the maximum defined in <i>parm2</i> of <b>controlcode</b> 132, only the maximum number of copies is printed</p> <p><b>FDEVICECONTROL</b> accepts page length values that are in multiples of 0.25 inches, but the Hewlett-Packard 268x printer is able to produce only pages that are multiples of 0.5 inches. For this reason, use only even values in <i>parm1</i> (8:8) (bit (15:1)=0).</p>
134	I16V	<p>Download/delete character set:</p> <p><i>parm1</i> (0:1)      0 = Download character set identified in right-hand byte of <i>parm2</i>, Hewlett-Packard 268x                               1 = Purge character set identified in right-hand byte of <i>parm2</i>, Hewlett-Packard 268x</p> <p><i>parm2</i> (0:1)      0 = First record of a load                               1 = Continuation of previous record</p> <p><i>parm2</i> (8:8)      Character set identifier; integer from 0 to 31</p> <p>If there is an attempt to download a character set having the same identifier as one existing in the printer, then the Hewlett-Packard 268x purges the existing character set and repacks the user area before loading the new font. However, before the modification of the user area, the Hewlett-Packard 268x prints all data currently in its buffer, as it does when a character set, form, or vertical format control set (VFC) is loaded, overlaid, or deleted.</p>

Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
135	I16V	<p>Download/delete form:</p> <p><i>parm1</i> (0:1)      0 = Load form set identified in the right-hand byte of <i>parm2</i>  1 = Purge identified form set from the Hewlett-Packard 268x printer's memory</p> <p><i>parm2</i> (0:1)      0 = First record of a load  1 = Continuation of previous record</p> <p><i>parm2</i> (8:8)      Form set identifier; integer from 0 to 31</p> <p>If attempting to download a form set having the same identifier as one existing in the printer, then the Hewlett-Packard 268x purges the existing form set and repacks the user area before loading the new form. However, before the modification of the user area, the Hewlett-Packard 268x prints all data currently in its buffer, as it does whenever you load, overlay, or delete a character set, form, or vertical format control set (VFC).</p>
136	I16V	<p>Download logical page table:</p> <p><i>parm1</i>              Not used</p> <p><i>parm2</i> (0:1)      0 = First record of a load  1 = Continuation of previous record</p> <p>A logical page is a page of data that may take up an entire sheet of paper. It is possible to print up to eight logical pages on one physical page. The logical page table, 513 words long, contains some of the information needed to print up to 32 logical pages, so that the set of up to eight logical pages printed on any one physical page can be varied.</p>
137	I16V	<p>Download multicopy form overlay table:</p> <p><i>parm1</i>              Not used</p> <p><i>parm2</i>              Not used</p> <p>Prints up to eight copies of a page, each on one or two different forms. FDEVICECONTROL downloads into the printer's memory a table containing one word of information for each of the eight possible copies to be overlaid with a form. The format of each word of the table is:</p> <p>Bit (0:1)            0 = Ignored  1 = Overlay form1 on physical page</p> <p>Bit (1:1)            0 = Ignored  1 = Overlay form2 on physical page</p> <p>Bits (2:4)            Reserved for the operating system</p> <p>Bits (6:5)            Form1 identifier; integer from 0 to 31</p> <p>Bits (11:5)          Form2 identifier; integer from 0 to 31</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1,Parm2 Description
138	I16V	<p>Download/delete vertical format control (VFC);</p> <p><i>parm1</i> (0:1)      0 = Load VFC                          1 = Delete VFC</p> <p><i>parm2</i> (0:1)      0 = First record of a load                          1 = Logical continuation of the previous record</p> <p><i>parm2</i> (8:8)      VFC set identifier; integer from 0 to 31</p> <p>The VFC table is an ASCII file downloaded to the Hewlett-Packard 268x printer in order to give specific instructions on the print density, location of the top of the page, the bottom of the page, and other specifications of the printed page.</p> <p>The Hewlett-Packard 268x expresses the height of a printed line in dots and the system uses this value to compute line positions on the page. Because these space measurements are relative to the top of the logical page, as opposed to the physical page, use the same or different VFC tables for logical pages of different rotations.</p>
139	I16V	<p>Download/delete a picture:</p> <p><i>parm1</i> (0:1)      0 = Load picture                          1 = Delete picture</p> <p><i>parm2</i> (0:1)      0 = First record of a load                          1 = Logical continuation of the previous record</p> <p><i>parm2</i> (8:8)      Picture identifier; integer from 0 to 31</p> <p>When the picture is downloaded to the Hewlett-Packard 268x, it changes every pointer to reflect where the dot per bit symbol actually is in memory. If a picture is downloaded and one is already present with the same identifier (0-31), then the original one is overwritten in the picture descriptor block (PDB). The area taken up by the deleted picture is freed as soon as the page has been transferred to paper.</p>
140	I16V	<p>Page control:</p> <p><i>parm1</i> (15:1)      0 = Do not eject physical page                          1 = Eject physical page before going to logical page (no effect if the first record since an environment load, <b>FOPEN/FCLOSE</b>)</p> <p><i>parm1</i> (13:2)      Auto-eject mode:</p> <p>                         00 = Use auto-eject flag of last data record (default at start; auto-eject enabled)                          01 = Enable auto-eject (select VFC channel 1 on new page)                          11 = Disable auto-eject (position pen at top of page)</p> <p><i>parm2</i> (8:8)      Logical page number; integer from 0 to 31</p> <p>The logical page identified in <i>parm2</i> becomes the current logical page even if other logical pages have entries that precede it in the logical page table. <b>FDEVICECONTROL</b> activates the specified page if it is inactive, and the Hewlett-Packard 268x performs a physical page eject if <i>parm1</i> (15:1)=1.</p>

Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
141	I16V	<p>Clear environment:</p> <p>When set to zero, the following bits are ignored.</p> <p><i>parm1</i> (0:1)      1 = Clear all character sets</p> <p><i>parm1</i> (1:1)      1 = Clear all forms</p> <p><i>parm1</i> (2:1)      1 = Clear all vertical format controls (VFCs)</p> <p><i>parm1</i> (3:1)      1 = Clear all pictures</p> <p><i>parm2</i>              Not used</p> <p>The printer flushes all data currently in its buffers and then performs the indicated clears, if any.</p>
142	I16V	Reserved for the operating system
143	I16V	<p>Load default environment:</p> <p><i>parm1</i>              Not used</p> <p><i>parm2</i>              Not used</p> <p>The Hewlett-Packard 268x printer flushes all data, erases the user area, and loads the default character set, the vertical format control (VFC), and the logical page table (LPT).</p>
144	I16V	<p>Print picture:</p> <p><i>parm1</i> (0:1)      0 = Temporary picture                          1 = Addressable picture</p> <p><i>parm1</i> (1:1)      0 = X and Y relative to the current pen position                          1 = X and Y are absolute pen position to the logical page</p> <p><i>parm1</i> (2:14)      X coordinate for picture placement (radixed integer)</p> <p><i>parm2</i> (0:1)      0 = First temporary picture load                          1 = Continuation record for load</p> <p><i>parm2</i> (1:1)      Not used</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1, Parm2 Description
145	I16V	End of job: <i>parm1</i> Not used <i>parm2</i> Not used
146	I16V	Device extended capability mode: <i>parm1</i> 0 = Clear 1 = Set <i>parm2</i> Not used
148	I16V	Device Status: Enable, disable, or return various types of extended device status. In order to maintain backward compatibility with MPE V/E, the initial setting (after first <b>FOPEN</b> of device) is set to disable all status reporting. Any application that requires status reporting must enable status reporting after opening the device. Status reporting is a global device feature, either every accessor sees it or does not see it. When enabled, device related I/O errors (if they occur) are not returned. The caller is notified that the device has status and another <b>FDEVICECONTROL</b> call has to be made to determine the status. When enabling status, the buffer returns the number of bytes occupied by the largest status block returned by the device. This number is returned in a 32-bit integer, and the count must be set to at least 4 bytes; otherwise, a bounds violation occurs. When nowait I/O is enabled, <b>FWRITE</b> does not notify the caller that status is available. The application is notified that the status is available after calling <b>IOWAIT</b> or <b>IODONTWAIT</b> . The status return can be either a wildcard (return any available status) or a specific type: <i>parm1</i> 0 Any available and enabled status 1 Device status 2 Environment status 3 Job status 4 Job error log status <i>parm2</i> 0 Return last buffered status 1 Return new actual status 2 Enable status specified in <i>parm1</i> 3 Disable status specified in <i>parm1</i>

Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
149	I16V	<p>Set data block number:</p> <p>To be used for printer error recovery. The data block number is used as the printing application/utility data index. The printing application can set data block numbers allowing it to return to that place in the data stream and retransmit data from that point forward.</p> <p><i>parm1</i>            Lower half of the 32-bit integer</p> <p><i>parm2</i>            Upper half of the 32-bit integer</p>
150	I16V	<p>Start silent run:</p> <p>Used to start printer recovery from specific error conditions on recoverable devices or to backup or skip forward a specified number of pages and resume printing.</p> <p><i>parm1</i>            Start silent run physical page number</p> <p><i>parm2</i>            Resume printing of physical page number</p>
151	I16V	<p>Print standard header or trailer:</p> <p>Used to print standard headers/trailers. If headers/trailers are disabled through the <b>BANNERS</b>, <b>HEADON</b>, or <b>HEADOFF</b> commands, this control code is invalid.</p> <p>PM (privileged mode) is required to use this option.</p> <p>The file system does not perform printer setup/reset functions before or after printing the banner; therefore, the caller must place the printer and paper in the proper state before and after calling this control code.</p> <p><i>parm1</i>            0   Print standard header</p> <p>                    1   Print standard trailer</p> <p><i>parm2</i>            Not used, must be set to zero</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1,Parm2 Description
192	U16V	<p>Perform control functions on terminal device files:</p> <p><i>parm1</i> Indicates the specific control directive applied to the terminal device file being accessed. For printers, only <i>parm1</i>=1 is supported. Valid values are:</p> <ol style="list-style-type: none"> <li>1 Specify terminal type or printer type file:            Specifies a new terminal type or printer type file for use with a device. The specified file may be created through the TTUTIL utility. The name of the terminal type or printer type file is passed in <i>buffer</i>, with the length of <i>buffer</i> set through the <i>length</i> parameter. If this call is issued against a device connected through PAD, CCE (2) is returned and the terminal or printer type associated with the device is changed, but the device operates as if no device control action took place.</li> <li>2 Set read timeout value for next read:            Sets a timeout value to be applied to the next read request in seconds. If the timer expires before the next read is terminated, the read data is transferred to the buffer and the read reply returns a software timeout error status with <i>length</i>=0. For reads posted to devices connected through PAD, add enough time to compensate for delays caused by the network.</li> <li>3 Set line speed for the device:            Set the line speed for the device associated with <i>filenum</i> by entering a new line speed, representing characters per second, as the value of <i>buffer</i>. Supported speeds are 30, 120, 240, 480, 960, and 1920 characters per second. Input and output line speeds are the same. If this call is issued against a device connected through PAD, the terminal control action is ignored and a meaningless value is returned.</li> <li>4 Set echo ON/OFF at terminal:            Enable or disable read data echo to a terminal through the DTC. Valid values are:            0 Echo off            1 Echo on             This call is not valid if the terminal is operating in binary or block mode.</li> </ol>



Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 5 Set system break response on or off at a terminal: Cont'd Enable or disable system response to the entry of a system <b>BREAK</b> at the terminal. Valid values are:</p> <ul style="list-style-type: none"> <li>0 Disable break</li> <li>1 Enable break</li> </ul> <p>6 Set subsystem break response on or off at a terminal: Enable or disable system response to the entry of a subsystem break character at the terminal. Valid values are:</p> <ul style="list-style-type: none"> <li>0 Disable subsystem break</li> <li>1 Enable subsystem break</li> </ul> <p>If the terminal is connected through PAD, the subsystem break must be followed by a <b>RETURN</b> to signal PAD that data is to be forwarded.</p> <p>8 Obtain time used for completion of last read: Determine the time, in seconds, required for the last read to complete. Ensure that this call is placed immediately after the read to be timed, with no subsequent reads in between. Only read access is allowed for this request. All reads are timed, so there is no need to enable the timer.</p> <p>9,11 Set parity generation and checking on or off: Enable or disable parity generation and checking between the driver and device. Valid values are:</p> <ul style="list-style-type: none"> <li>0 Disable parity; all eight bits of each character are passed through.</li> <li>1 Enable parity; type of parity specified by the parity setting is in effect.</li> </ul> <p>Input and output parity is the same. If issued for a terminal connected through PAD, CCE (2) is returned, but no terminal control action occurs.</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1, Parm2 Values (continued)**

Control-code	Mnemonic	Parm1, Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 10, Set type of parity:</p> <p>Cont'd 12 If parity is enabled, this request determines what kind of parity is computed when parity is checked:</p> <ul style="list-style-type: none"> <li>0 No parity check on input, eighth bit set to 0 on output (parity type = zeros)</li> <li>1 No parity check on input, eighth bit set to 1 on output (parity type = ones)</li> <li>2 Even parity checked on input, even parity generated on output (parity type = even)</li> <li>3 Odd parity checked on input, odd parity generated on output (parity type = odd)</li> </ul> <p>If parity is disabled, no parity is used.</p> <p>If issued for a terminal connected through PAD, CCE (2) is returned, but no terminal control action occurs.</p> <p>14 Set line deletion response:</p> <p>Enables or disables the transmission of line deletion characters (!!!) to the terminal after a user cancels a line at a terminal:</p> <ul style="list-style-type: none"> <li>0 Disable transmission</li> <li>1 Enable transmission</li> </ul> <p>Line deletion characters are not output by PAD devices. If issued to a terminal connected through PAD, CCE (2) is returned, but no terminal control action occurs.</p>

Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 15 Set transparent editing mode:</p> <p>Cont'd Enables or disables transparent (unedited) mode. When enabled, allows most special characters to be read and treated as data and a small subset to retain their meaning:</p> <p>AEOR The AEOR character defined through a previous call to <b>FCONTROL</b> <i>itemnum</i>=25. The read terminates in error, and a call to <b>FCHECK</b> returns an error code of 31. The AEOR character is passed with the data and included in the byte count.</p> <p><b>BREAK</b> Causes a system break (when enabled) and transfers control to the system. If the <b>RESUME</b> command is entered, transparent editing is resumed.</p> <p><b>CTRL</b>Q (DC1 or XON) The resume output character of the XON/XOFF handshake. This character is stripped from the input data when XON/XOFF protocol is enabled.</p> <p>DC2 When input as the first character, it is stripped out. A DC1 (<b>CTRL</b>Q) is written to the terminal. If not input as the first character, DC2 is a data character.</p> <p><b>CTRL</b>S (DC3 or XOFF) The stop output character of the XON/XOFF handshake. This character is stripped from the input data when XON/XOFF protocol is enabled.</p> <p>Enable transparent editing by passing non-NUL characters for the subsystem break and EOR characters in the buffer in the following format:</p> <p>bits Specifies the character to be used as the subsystem break (0:8) character. Any character not defined as a special character in transparent mode (including the standard subsystem break character) can be used. If set to 0, subsystem break is disabled.</p> <p>bits Specifies the character to be used as the line termination (8:8) character (EOR). Any character not defined as a special character in transparent mode can be used.</p> <p>Disable transparent editing by passing NUL characters.</p> <p>If used for a device connected through PAD, data is not forwarded from the terminal immediately after the characters defined as the subsystem break or EOR character are typed. The terminal user has to press <b>RETURN</b> to cause data to be transmitted, or wait for the data forwarding timer to expire (if set). The host software recognizes and acts on the characters defined in this call.</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1,Parm2 Description
192 Cont'd	U16v	<p><i>parm1</i> 26 Device XON/XOFF:</p> <p>Enable or disable XON/XOFF flow control between the device and the DTC:</p> <p>1 Enable device XON/XOFF</p> <p>0 Disable device XON/XOFF</p> <p>When enabled, the DTC stops sending data to the device when XOFF is received and resume when XON is received. XON/XOFF characters are not sent to the host.</p> <p>When disabled, the XON/XOFF characters are not treated as control characters, but are passed to the host as data. When disabled, data overruns can occur.</p> <p>27 Set XOFF timer value:</p> <p>Enable or disable the XOFF timer.</p> <p>To enable the XOFF timer, pass a positive value, representing time in seconds, in the <i>buffer</i> parameter. The XOFF timer causes a warning message to be sent to the console if a device is XOFFed for a time exceeding the set limit.</p> <p>To disable the XOFF timer, pass a 0 or negative value in the <i>buffer</i> parameter.</p> <p>If issued to a terminal connected through PAD, CCE (2) is returned, but no terminal control action occurs.</p> <p>28 Block mode types supported:</p> <p>This is a read only request that returns a value representing the type of block mode supported by the driver. Valid values are:</p> <p>6 DTC style page block mode</p> <p>7 Line and DTC style page block mode</p> <p>15 PAD terminal supporting page block mode</p>

Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 29 Block mode alert character: Cont'd Specifies the character used to signal Hewlett-Packard block mode transfers. The standard alert character is DC2. This call only defines the alert character; it does not enable Hewlett-Packard block mode.</p> <p>If issued to a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p>30, Read trigger and block mode trigger character: 32 Specify which character the DTC should send to trigger a character mode or block mode read on a device. The standard trigger character is DC1. If a NUL character is specified, there is no trigger character.</p> <p>If issued to a terminal connected through PAD, CCE (2) is returned but the device control action does not occur.</p> <p>36 Backspace character: Specifies which character will be a backspace in standard editing mode. Default: backspace</p> <p>37 Cancel line character Specifies which character will be a line deletion character in standard editing mode. Default: <b>CTRL</b>X</p> <p>39 Type 1 EOR character: Specifies the EOR character to be used in standard or transparent editing mode. This value is passed in the high order byte (value * 256) of <i>buffer</i>. If issued to a terminal connected through a PAD, the new EOR character (non-carriage return) must be followed by a carriage return to instruct the PAD to transmit the data. Use a null character to disable the character. Default: <b>Return</b></p> <p>40 Type 2 EOR character (AEOR): Specifies the AEOR character and enables its use. This value is passed in the high order byte (value * 256) of <i>buffer</i>. Any character can be specified, but do not use any special characters (refer to <b>FCONTROL itemnum=25</b> for details). If issued to a terminal connected through a PAD, the new EOR character (non-carriage return) must be followed by a carriage return to instruct the PAD to transmit the data. Use a null character to disable the AEOR character.</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1,Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 41 Subsystem break character: Cont'd</p> <p>Specifies the character is to be used as the subsystem break character. This value is passed in the high order byte (value * 256) of <i>buffer</i>.</p> <p>If issued to a terminal connected through PAD, the new subsystem break character is recognized by the host, but the terminal user has to press <b>RETURN</b> following the subsystem break to signal the PAD that data is ready to be transmitted.</p> <p>51 Set typeahead mode: Enable or disable typeahead mode:</p> <p>0 Disable typeahead mode; accept read data from the device as it is posted. The terminal user must wait for a new read to be posted before entering additional data. (default)</p> <p>1 Enable typeahead mode; pass read data from the device to a typeahead buffer (up to 224 characters). The terminal user can enter data before a read has been posted. Typeahead is not supported on PAD terminals. If issued for a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p>52 Form feed in output data: Specifies whether the form feed character is replaced in the carriage control of terminals:</p> <p>0 Replace form feeds with a form feed replacement character</p> <p>1 Form feed allowed</p> <p>53 Define form feed replacement character: Specifies the character that replaces the form feed character in the terminal carriage control when form feed replacement is enabled. The form feed in carriage control is replaced; form feed in data is not replaced. A default of NUL is used if form feed replacement is enabled and no replacement character is specified.</p>

Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 55 Backspace response action: Cont'd Selects the action the DTC takes when it receives a backspace character. Valid values are:</p> <ul style="list-style-type: none"> <li>1 Remove character from input and back cursor up one space (default)</li> <li>5 Remove character from input and erase character (backspace, space, backspace)</li> </ul> <p>If issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p>56 Data bits per character: Specifies whether 7-bits or 8-bits are used for each character. If 7-bits is selected, parity generation and checking is controlled by the current parity setting. If the current parity is set to “no parity” or parity checking is disabled, parity is not in effect. If 8-bits is selected, parity is set to “no parity” and the current parity setting is saved, so when the user switches back to 7-bits, parity is returned to its previous setting. If issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p>57 Subsystem break character: Returns the current subsystem break character.</p> <p>60 Flush typeahead buffer: Valid values are:</p> <ul style="list-style-type: none"> <li>1 Flush the typeahead buffer before the next read is started. Any data in the typeahead buffer is lost. Valid for the next read only.</li> <li>0 Disable the request.</li> </ul> <p>Flush typeahead is enabled only if typeahead is enabled. If issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p>

**FDEVICECONTROL**

**Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)**

Control-code	Mnemonic	Parm1, Parm2 Description
192 Cont'd	U16V	<p><i>parm1</i> 61 Bypass typeahead buffer: Valid values are:</p> <ul style="list-style-type: none"> <li>1 The next read is taken from the device, and any information currently in the typeahead buffer is bypassed. The data in the typeahead buffer is not flushed, and can be obtained by a subsequent read. The request is valid for the next read only.</li> <li>0 Disable the request.</li> </ul> <p>This request is valid only if typeahead is enabled.</p> <p>If issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p>62 Quiesce I/O: Valid values are:</p> <ul style="list-style-type: none"> <li>1 The control will not complete until all previous I/O to the device has completed. If issued against a terminal connected through a PAD, control issues only previous I/O sent to PAD and not to the device.</li> </ul> <p>63 Single echo mode in typeahead: Valid values are:</p> <ul style="list-style-type: none"> <li>1 Enables single echo. Typeahead data will be echoed only when the read is posted.</li> <li>0 Disable the request. Typeahead data will be echoed when entered and when the read is posted (default mode).</li> </ul> <p>This request is valid only if typeahead is enabled.</p> <p>If issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p>64 Ignore parity error on input: Valid values are:</p> <ul style="list-style-type: none"> <li>1 Parity errors will be ignored.</li> <li>0 Parity errors will be reported (default).</li> </ul> <p>If issued against a terminal connected through PAD, CCE (2) is returned, but the device control action does not occur.</p> <p><i>parm2</i> Indicates the read and write access for the operation being performed. Set <i>parm2</i> to one of the following values:</p> <ul style="list-style-type: none"> <li>1 Read only; return current information</li> <li>2 Write only; pass new information</li> <li>3 Read/Write; pass new information and return valid information before call</li> </ul> <p>Information is returned in the area pointed to by the <i>buffer</i> parameter. For detailed information, refer to the <i>Asynchronous Serial Communications Programmer's Reference Manual (32022-90002)</i>.</p>



Table 4-4. FDEVICECONTROL Controlcode/Parm1,Parm2 Values (continued)

Control-code	Mnemonic	Parm1,Parm2 Description
193	I16V	<p>Record processing information for NRJE spoolfiles:</p> <p><i>parm1</i>                    Indicates the character code (for example, ASCII) of spoolfile data. Refer to <i>SNA NRJE Network Remote Job Entry User/Programmer Reference Manual</i> (30245-90001).</p> <p><i>parm2</i> (14:1)            0 = Data not compacted                                  1 = Data compacted</p> <p><i>parm2</i> (15:1)            0 = Data not compressed                                  1 = Data compressed</p>

**FDEVICECONTROL****Table 4-5. FDEVICECONTROL Controlcode 192 Values**

<b>Parm1</b>	<b>Parm2</b>	<b>Description</b>
1	R/W	Specify terminal or printer type file
2	R/W	Set read timeout value for the next read
3	R/W	Set the line speed for the device
4	R/W	Set echo ON/OFF at a terminal
5	R/W	Set system break response ON/OFF at a terminal
6	R/W	Set subsystem break response ON/OFF at a terminal
8	R	Obtain time used for completion of the last read
9	R/W	Set parity checking ON/OFF
10	R/W	Set type of parity to check
11	R/W	Set parity generation ON/OFF
12	R/W	Set type of parity generated
14	R/W	Set line deletion response
15	R/W	Set transparent editing mode
26	R/W	Device XON/XOFF enable
27	R/W	Set XOFF timer value
28	R	Block mode types supported
29	R/W	Define block mode alert character
30	R/W	Define block mode trigger character
32	R/W	Define read trigger character
36	R/W	Define backspace character
37	R/W	Define cancel line character
39	R/W	Define type 1 EOR character
40	R/W	Define type 2 EOR character (AEOR)

Table 4-5. FDEVICECONTROL Controlcode 192 Values (continued)

Parm1	Parm2	Description
41	R/W	Define subsystem break character
51	R/W	Set type ahead mode
52	R/W	Form feed allowed in output data
53	R/W	Define form feed replacement character
55	R/W	Select backspace response action
56	R/W	Specify data bits per character
57	R	Obtain subsystem break character
60	R/W	Flush typeahead buffer
61	R/W	Bypass typeahead buffer
62	W	Quiesce I/O
63	R/W	Single echo mode typeahead
64	R/W	Ignore parity error

### Operation Notes

Not used for KSAM files.

### Condition Codes

CCE (2) Request granted.  
 CCG (0) Not returned.  
 CCL (1) Request denied. An error occurred.

### Related Information

Intrinsics FCONTROL

Commands None

Manuals *Accessing Files Programmer's Guide* (32650-90017), *IFS/3000 Reference* (36580-90001), and *MPE XL Asynchronous Serial Communications Programmer's Reference Manual* (32022-90012)

---

## FERRMSG

NM and CM callable.

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

### Syntax

I16	CA	I16
FERRMSG( <i>ferrorcode</i> , <i>msgbuffer</i> , <i>msglength</i> );		

### Parameters

- ferrorcode*    **16-bit signed integer by reference (required)**  
Passes an error code returned by the FCHECK intrinsic indicating which message to return in *msgbuffer*.
- msgbuffer*    **character array (required)**  
Returns the error message identified with *ferrorcode*. To contain the longest possible message, *msgbuffer* must be  $\geq 72$  bytes long.
- msglength*    **16-bit signed integer by reference (required)**  
Returns the length of the error message in *msgbuffer*. The length is returned in positive bytes.

### Condition Codes

- CCE (2)        Request granted.
- CCG (0)        Request denied. No error message exists for this *ferrorcode*.
- CCL (1)        Request denied. The *msgbuffer* address was out of bounds, *msgbuffer* was not large enough, or *msglength* was out of bounds.

### Related Information

- Intrinsics     None
- Commands      None
- Manuals        *Accessing Files Programmer's Guide* (32650-90017).

---

## FFILEINFO

NM and CM callable.

Returns information about a file.

### Syntax

I16V          I16V    *
FFILEINFO( <i>filenum</i> [, <i>itemnum</i> , <i>item</i> ] [ . . . ] );

---

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

---

### Parameters

<i>filenum</i>	<b>16-bit signed integer by value (required)</b> Passes the file number of the file for which information is requested.
<i>itemnum</i>	<b>16-bit signed integer by value (optional)</b> Specifies which <i>item</i> value is to be returned. (Refer to Table 4-6.)
<i>item</i>	<b>type varies (optional)</b> Returns the value of the item specified in the corresponding <i>itemnum</i> . (Refer to Table 4-6.)

Table 4-6. FFILEINFO Itemnum/Item Values

Itemnum	Item Type	Item Description
1	CA	<p>File designator (28 bytes):</p> <p>Returns the file designator of the file being referenced in the format:  <i>filename.groupname.accountname</i>            Must be <math>\geq 28</math> bytes in length. Unused bytes are filled with right-justified blanks and a nameless file returns an empty string. The fully qualified name of the file referenced by <i>filenum</i> is returned as the value of this <i>itemnum</i>. Only names which can be expressed using MPE-only semantics are returned by this <i>itemnum</i>. If the name of the object referenced by <i>filenum</i> can not be expressed using MPE-name semantics a CCL condition code is returned. Calling <b>FCHECK</b> for <i>filenum</i> after this error occurs will return <b>FSERR391</b>.</p>
2	U16	<p>File options:</p> <p>Returns file characteristics (refer to the <i>foption</i> figure). The record format extension bit is returned as the <i>foption</i> (1:1) bit. Byte stream record format is represented as a record format extension of one with a variable record format <i>foption</i> (8:2) bits equal to 01.</p> <p>Directories, symbolic links, device links, pipes and FIFO's can not be represented by <i>foptions</i>. If the object referenced by <i>filenum</i> is one of these objects, a CCL condition code is returned. Calling <b>FCHECK</b> for <i>filenum</i> after this error occurs will return <b>FSERR399</b></p>
3	U16	<p>Access options:</p> <p>Returns file access information (refer to the <i>Faoption</i> figure).</p>
4	I16	<p>(CM) Record size: Returns the logical record size associated with the file:</p> <ul style="list-style-type: none"> <li>■ If the file was created as a binary file, this value is positive and is in half words.</li> <li>■ If the file was created as an ASCII file, this value is negative and is in bytes.</li> </ul> <p>For message files, when there is call to <b>FCONTROL</b> with <i>controlcode</i>=46, the value returned is the size of the data records, including the 4 byte header.</p> <p>Maintained for compatibility with MPE V/E-based systems only. CM record sizes are imposed when <b>FGETINFO</b> returns record size information on all file types. If the record size exceeds the limits, a zero is returned.</p> <p>Note: If a zero is returned, use item 67.</p>
5	I16	<p>Device type/subtype:</p> <p>Returns the type and subtype of the device being used for a KSAM, RIO, circular, or message file, or devices such as a tape drive, printer, or terminal where bits (0:8) indicate the device subtype, and bits (8:8) indicate the device type.</p> <p>If the file is not spooled or is opened as a spoolfile through the logical device, the actual value is returned. If an output file is spooled and was opened by device class name, the type and subtype of the first device in its class is returned. (This may be different from the device actually used.)</p>

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description
6	U16	<p>Logical device number:</p> <p>Returns the logical device number of the device where the disk file label resides.</p> <ul style="list-style-type: none"> <li>■ If the file is a disk file, the LDEV is the location of the file label. (File data can reside on the same device as the file label.)</li> <li>■ If the file is spooled, the LDEV is a virtual device number that does not correspond to the system configuration I/O device list.</li> <li>■ If the file is located on a remote computer, linked by a DS point-to-point or X.25 link, the left eight bits (0:8) are the LDEV of the distributed system (DS) device.</li> <li>■ If the file is located on a remote computer, linked by NS 3000/XL, the left eight bits (0:8) are the remote environment of the connection. The right eight bits (8:8) are the LDEV of the device on the remote computer where the file label resides.</li> <li>■ If the DS device for the RFA or the LDEV is 0, then a zero is returned.</li> </ul> <p>Note: If a zero is returned, use item 50.</p>
7	U16	<p>Hardware device address:</p> <p>Returns 2048. Maintained to provide backward compatibility with MPE V/E-based systems.</p>
8	I16	<p>File code:</p> <p>Returns the file code of a disk file (refer to <b>FFILEINFO</b> for file codes).</p>
9	I32	<p>Current logical record pointer:</p> <p>Returns the current logical record pointer setting. This value is the displacement in logical records from record number 0 in the file and identifies the record that would be accessed next by <b>FREAD</b> or <b>FWRITE</b>.</p>
10	I32	<p>EOF:</p> <p>Returns the pointer setting of the last logical record currently in the file (equivalent to EOF). If the file does not reside on disk, the value is zero. For message files, when a call is made to <b>FCONTROL</b> with <i>itemnum</i>=46, the number of records returned includes open, close, and data records.</p>
11	I32	<p>File limit:</p> <p>Returns a number representing the last logical record that can exist in the file (equivalent to the file limit). If the file does not reside on disk, the value is zero.</p>
12	I32	<p>Log count:</p> <p>Returns the logical records passed to and from the program during the current file access.</p>
13	I32	<p>Physical count:</p> <p>Returns the number of buffered physical I/O operations performed since the last <b>FOPEN/HPFOPEN</b> call (records).</p>

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description
14	I16	<p>Block size:</p> <p>Returns the file block size:</p> <ul style="list-style-type: none"> <li>■ If the file is binary, the value is positive and the size is in half words.</li> <li>■ If the file is ASCII, the value is negative and the size is in bytes.</li> </ul> <p>For standard fixed ASCII files with odd record sizes, the block size is computed as “(record size + 1)” X block factor. For even record sizes, it returns the block size from the gufd.</p> <p>Maintained for compatibility with MPE V/E-based systems only. CM block size limits are used when <b>FGETINFO</b> returns block size information on all file types (STD, KSAM, RIO, CIR, MSG). If the block size of the specified file exceeds the limits, zero is returned.</p> <p>Note: If a zero is returned, use item 68.</p>
15	I16	<p>Extent size:</p> <p>Returns the extent size; for compatibility with MPE V/E-based systems only.</p> <p>Note: If a zero is returned, use item 69. If extent size is specified or the maximum number of extents is specified at file creation, the size and number of extents are determined by the operating system and the <i>item</i> values are not actual values; they are calculated using system defaults.</p>
16	U16	<p>Maximum number of extents:</p> <p>If the extent size or maximum number of extents is specified as zero at file creation, then the size and number of extents are determined by the system. In that case, these item values are calculated using system defaults defaults and do not reflect actual values.</p>
17	I16	<p>User labels:</p> <p>Returns the number of user labels defined for the file during creation. If the file is not a disk file, this number is zero. When an old file is opened for overwrite output, the value is not reset and the old user label is not destroyed.</p>
18	CA	<p>Creator:</p> <p>Returns the name of the file creator (at least 8 bytes). If the file does not reside on disk, blanks are returned.</p> <p>An unqualified form of the file owner’s name is returned as the value of this <i>itemnum</i>. The file owner is not necessarily the file’s creator. File ownership may be changed using the <b>:ALTFILE</b> command.</p> <p>A symbolic zero (ASCII 48 in decimal) is returned as the file owner for root directories, accounts, and MPE groups created prior to release 4.5 of MPE/iX.</p> <p>If the file is not located in the account in which the file owner is a member, a blank file owner name is returned. Item number 85 should be used to obtain the full file owner name instead of item 18.</p>



Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description
19	I32	Label address: Returns a zero. For compatibility with MPE V/E-based systems only.
20	I16	Blocking factor
21	I16	Physical block size; indicates half words
22	I16	Data block size; indicates half words
23	I16	Offset to data in blocks; indicates half words
24	I16	Offset of active record table for RIO files; indicates half words
25	I16	Size of active record table within the block; indicates half words
26	CA	Volume ID (tape label)
27	CA	Volume set ID (tape label)
28	U16	Expiration date (calendar format)
29	I16	File sequence number
30	I16	Reel number
31	I16	Sequence type
32	U16	Creation date (calendar format)
33	I16	Label type
34	I16	Current number of writers
35	I16	Current number of readers
36	U16	File allocation date, when the file was last restored ( <b>CALENDAR</b> format)
37	I32	File allocation time, when the file was last restored ( <b>CLOCK</b> format)
38	U16	Spoolfile device file number: Bits (1:15) = Device file number Bit (0:1) = 1 Output spoolfile Bit (0:1) = 0 Input spoolfile  If the spoolfile device number is larger than 32767, <i>itemnum</i> 38 returns 0 (zero). Use <i>itemnum</i> 78 instead for spoolfile numbers larger than 32767.
40	I32	Disk device status: Returns a zero. For compatibility with MPE V/E-based systems only.

**FFILEINFO**

**Table 4-6. FFILEINFO Itemnum/Item Values (continued)**

Itemnum	Item Type	Item Description
41	I16	Device type
42	I16	Device subtype
43	CA	Environment file name (>=36 bytes)
44	I16	Number of disk extents currently allocated to the file
45	CA	File name from labeled tape header 1 record (>= 17 bytes)
46	I16	Tape density
47	I16	DRT number: Always returns an 8.
48	I16	Device unit number: Always returns a 0.
49	U16	Equivalent to a software interrupt PLABEL for message files
50	U16	Real device number of the file
51	I16	Remote environment number  Note: If using NS 3000/XL RFA (remote file access), specify <i>DSDEVICE ldev#</i> when you are using a DS (point-to-point or X.25) link.
52	I32	Last modification time ( <b>CLOCK</b> format)  Zero is returned as the modification time for root directories, accounts, and MPE groups created prior to release 4.5.
53	U16	Last modification date ( <b>CALENDAR</b> format)  Zero is returned as the modification time for root directories, accounts, and MPE groups created prior to release 4.5.
54	U16	File creation date ( <b>CALENDAR</b> format)  Zero is returned as the modification time for root directories, accounts, and MPE groups created prior to release 4.5.
55	U16	Last access date ( <b>CALENDAR</b> format)  Zero is returned as the modification time for root directories, accounts, and MPE groups created prior to release 4.5.

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description
56	I32	Number of data blocks in a variable length file
57	I16	Number of user labels written to the file
58	I16	Number of accessors having output access (write) for a particular file
59	I16	Number of accessors having input access (read/update) for a particular file
60	I16	Terminal type: 0 File's associated device not a terminal 1 Standard hardwire or multipoint terminal 2 Terminal connected through phone-modem 3 DS pseudo-terminal 4 X.25 Packed Switching Network PAD (packet assembler/disassembler) terminal 5 NS virtual terminal
61	CA	NS 3000/XL remote environment ID name  Note: If using NS 3000/XL RFA (remote file access), specify <code>DSDEVICE ldev#</code> when using a DS (point-to-point or X.25) link. A buffer must be provided for the node name (or <code>envid</code> ) with the required space of 52 bytes; otherwise, data corruption may occur on variables following <code>itemnum=61</code> or an <code>FSERR 73</code> , <code>BOUNDS VIOLATION</code> may be returned.
62	CA	File lockword (8 bytes):
63	CA	Unique file identifier (UFID) (20 bytes):
64	@64	Virtual address of the file:  Applicable for standard disk files only. (Requesting <code>itemnums</code> 64, 74, or 75 for any other file type, RIO, MSG, CIR, causes an error and returns CCL (1).)
65		Reserved for the operating system.
66	@32	Virtual address of global unique file descriptor (GUFD):

**FFILEINFO**

**Table 4-6. FFILEINFO Itemnum/Item Values (continued)**

Itemnum	Item Type	Item Description
67	U32	(NM) Record size (indicates bytes)
68	U32	Block size (indicates bytes). Odd or even record sizes are returned as the block size from the gufd.
69	U32	Extent size (indicates bytes)
74	@64	Virtual address of file label:  Applicable for standard disk files only. (Requesting <i>itemnums</i> 64, 74, or 75 for any other file type (RIO, MSG, CIR) causes an error and returns CCL (1).)
75	CA	Hardware path:  Applicable for standard disk files only. (Requesting <i>itemnums</i> 64, 74, or 75 for any other file type (RIO, MSG, CIR) causes an error and returns CCL (1).)
76	CA	Volume restriction (34 bytes):  The last two characters indicate the type:  0 File placed on the specified volume at creation 1 File can be placed on any volume containing the specified class at creation 2 File can be placed on any volume within the specified volume set at creation (Default)
77	U32	Transaction management log set ID  If <i>itemnum</i> 77 = 0 (zero), the file is not attached to the XM (Transaction Management) log.
78	U32	Spoolfile device file number:  Bits (1:31) = Device file number Bit (0:1) = 1 Output spoolfile Bit (0:1) = 0 Input spoolfile
79	I16	File's pending disposition  0 = No change, the disposition is the same as before the file was opened 1 = Permanent 2 = Temporary (tape files rewound) 3 = Temporary (same as 2 except tape files not rewound) 4 = Released (purged) 5 = Temporary (but the file was previously a permanent file)

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description
80	CA	<p>HFS syntax filename.</p> <p>This <i>itemnum</i> returns a null-terminated POSIX-syntax system absolute pathname for the file or directory referenced by <i>filenum</i>. On input the first four bytes of this buffer are interpreted as a 32-bit unsigned integer specifying the maximum buffer size in bytes. This maximum buffer size does not include the four bytes used to represent this size. On output the first four bytes of the buffer represent the pathname length excluding the null-terminator as an unsigned integer. The pathname is returned in the bytes following the pathname length. Bytes beyond the null-terminator should be considered undefined. If the maximum buffer length is incorrect on input, variables allocated near the buffer may be overwritten or a bounds violation may occur. A zero pathname length is returned for unnamed new files and when an error occurs. Zero is the minimum buffer length on input for this <i>itemnum</i>.</p> <p>Format of the buffer on input:</p> <pre> byte      #1  #2  #3  #4   #1  ...  #N +-----+-----+-----+-----+ ... +-----+             maximum                         length = N         +-----+-----+-----+-----+ ... +-----+ </pre> <p>Format of the buffer on output:</p> <pre> byte      #1  #2  #3  #4   #1  ...  #L  ...  #N +-----+-----+-----+-----+ ... +-----+ ... +-----+             length =             /               \0                           (L - 1)   +-----+-----+-----+-----+ ... +-----+ ... +-----+ </pre>
81	U32	The current number of hard links to the file.
82	I32	<p>Time of last file access in clock format. The bit assignments are:</p> <p>Bits 0 to 7        hours</p> <p>Bits 8 to 15      minutes</p> <p>Bits 16 to 23     seconds</p> <p>Bits 24 to 31     tenths of seconds</p>
83	I32	Time of last file status change in clock format (See item 82 for a description of clock format).
84	U16	<p>Date of last file status change in calendar format. The bit assignments are:</p> <p>Bits 0 - 7        Year of the century</p> <p>Bits 8 - 15       Day of the year</p>

**FFILEINFO****Table 4-6. FFILEINFO Itemnum/Item Values (continued)**

Itemnum	Item Type	Item Description
85	CA	File Owner The full file owner name. Unused characters are blank filled. A symbolic zero (ASCII 48 in decimal) is returned as the file owner for root directories, accounts, and MPE groups created prior to the POSIX release.
86	I32	File owner identifier The file owner identifier (UID). Zero is returned as the file owner ID for root directories, accounts, and MPE groups created prior to the POSIX release.
87	CA	File group The file group name. Unused characters are blank filled. A symbolic zero (ASCII 48 in decimal) is returned as the file group for root directories whose GID's have not been assigned.

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description
88	I32	File group identifier The file group identifier (GID). Zero is returned as the file group ID for root directories whose GID's have not been assigned.
89	U32	File type The following valid file types may be returned: 0            Ordinary File 1            KSAM/3000 2            RIO 3            KSAM XL 4            CIR 5            Native Mode Spool File 6            MSG 7            Not Applicable 8            Compatability mode file 9            Directory 10-11       Not Applicable 12          Pipe 13          FIFO 14          Symbolic link 15          Device link

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description																						
90	U32	<p>Record type</p> <p>The following valid record types may be returned:</p> <table data-bbox="456 415 922 919"> <tr><td>0</td><td>Fixed</td></tr> <tr><td>1</td><td>Variable</td></tr> <tr><td>2</td><td>Undefined</td></tr> <tr><td>3</td><td>Spool block</td></tr> <tr><td>4</td><td>Root directory</td></tr> <tr><td>5</td><td>Not applicable</td></tr> <tr><td>6</td><td>Account directory</td></tr> <tr><td>7</td><td>Group directory</td></tr> <tr><td>8</td><td>Not applicable</td></tr> <tr><td>9</td><td>Byte stream</td></tr> <tr><td>10</td><td>Hierarchical directory</td></tr> </table> <p>This item will return information based upon the record format that is logically associated with the specified <i>file designator</i> and will not necessarily be the same as the physical record structure of the file. It is possible with the <b>HPFOPEN</b> intrinsic item 77, <b>Read Format</b> option, to access a file in a different mode than the physical record type. For example, byte stream record files, when opened by default, present <b>variable</b> record instead of the physical byte stream storage, since the file was opened to behave like a variable record file. In order to determine the true physical record structure of a file, use the <b>FFILEINFO</b> item 101.</p>	0	Fixed	1	Variable	2	Undefined	3	Spool block	4	Root directory	5	Not applicable	6	Account directory	7	Group directory	8	Not applicable	9	Byte stream	10	Hierarchical directory
0	Fixed																							
1	Variable																							
2	Undefined																							
3	Spool block																							
4	Root directory																							
5	Not applicable																							
6	Account directory																							
7	Group directory																							
8	Not applicable																							
9	Byte stream																							
10	Hierarchical directory																							
91	I64	<p>The current file size in bytes.</p> <p>The value returned represents the current position of the End-of-File (EOF) and may not reflect the number of bytes actually occupied by the file on disk if the file is sparsely allocated.</p>																						
92	I32	<p>KSAM XL file version:</p> <p>This item returns a value indicating the version of a KSAM XL file. A value of 1 indicates an original type KSAM XL file, and a value of 2 indicates the next generation KSAM XL file. A value of zero is returned if the file is not a KSAM XL file.</p>																						
93	U32	<p>NM Plabel:</p> <p>This item returns a 32-bit NM Plabel of a message file interrupt handler. Interrupts may be enabled on message files by calling the <b>FCONTROL</b> intrinsic with item 48 and the Plabel address.</p>																						



Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description																		
94	I32	<p>MPE/iX device type:</p> <p>This item returns the following values for the following types of devices:</p> <table data-bbox="456 415 850 825"> <tr><td>0</td><td>Disk device</td></tr> <tr><td>1</td><td>Tape device</td></tr> <tr><td>2</td><td>Terminal device</td></tr> <tr><td>3</td><td>Printer device</td></tr> <tr><td>4</td><td>Remote device</td></tr> <tr><td>5</td><td>Ports device</td></tr> <tr><td>6</td><td>Reserved</td></tr> <tr><td>7</td><td>Streams device</td></tr> <tr><td>8</td><td>Sockets device</td></tr> </table>	0	Disk device	1	Tape device	2	Terminal device	3	Printer device	4	Remote device	5	Ports device	6	Reserved	7	Streams device	8	Sockets device
0	Disk device																			
1	Tape device																			
2	Terminal device																			
3	Printer device																			
4	Remote device																			
5	Ports device																			
6	Reserved																			
7	Streams device																			
8	Sockets device																			
95	I32	<p>Close-on-Exec</p> <p>This item returns a value indication whether or not this <i>filenum</i> is closed if one the POSIX.1 <code>exec()</code> family of functions is called. A value of 1 means that the file is closed on an <code>exec()</code> call, while a value of 0 indicates the file will survive across <code>exec()</code> calls.</p>																		
96	I32	<p>POSIX Append mode</p> <p>This item returns a value indicating whether or not this <i>filenum</i> has the POSIX.1 append mode flag set. When the append mode flag is set on files that support this feature, all writes occur at the end of the file, although reads may occur anywhere in the file. A value of 1 indicates that the POSIX.1 append mode is on, while a value of 0 indicates the append mode is off.</p> <p>The only time that the POSIX.1 append mode is valid is when a file has been opened for byte stream access (<code>HPFOPEN</code> option 77 with a value of 2).</p>																		
97	I32	<p>POSIX non-block mode</p> <p>This item returns a value indicating whether or not this <i>filenum</i> has the POSIX.1 non-block flag set. When the non-block flag is set, on files that support this feature, reads, writes, and opens can be affected in a file dependent manner. In general, operations that would otherwise have impeded the caller results in immediate return when this flag is set. A value of 1 indicates the non-block flag is set, while a value of zero indicates the flag is not set.</p> <p>The only time the non-block flag is valid is for pipes and FIFO's.</p>																		
98	I32	<p>Carriage Control</p> <p>This item returns a value indicating whether or not the carriage control option is in effect for this file. If carriage control is on, a value of 1 is returned. Otherwise, a value of 0 is returned.</p>																		

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description																						
99	I32	<p>Setuid Flag</p> <p>This item returns a value indicating whether or not this file has the setuid flag turned on. When the setuid flag is on for a program file, the program will execute under the user who owns the file rather than under the user who invoked the program. A value of 1 indicates that the setuid flag is on; a value of 0 indicates the flag is off.</p>																						
100	I32	<p>Setgid Flag</p> <p>This item returns a value indicating whether or not the file has the setgid flag turned on. When the setgid flag is on for a program file, the program will execute with the GID membership of the file rather than the GID membership of the process that invoked the program. A value of 1 indicates that the setgid flag is on; a value of 0 indicates that the flag is off.</p>																						
101	I32	<p>Physical record type</p> <p>This item returns the record type of the file's physical record layout. It is possible through the <b>HPFOPEN</b> option 77, <b>Read Format</b> option, to emulate a different record structure than the actual physical record structure of the file. To get the current logical record view, use <b>FFILEINFO</b> item 90.</p> <p>The following values will be returned by this item:</p> <table data-bbox="459 1014 914 1518"> <tr><td>0</td><td>fixed</td></tr> <tr><td>1</td><td>variable</td></tr> <tr><td>2</td><td>undefined</td></tr> <tr><td>3</td><td>spool block</td></tr> <tr><td>4</td><td>root directory</td></tr> <tr><td>5</td><td>not applicable</td></tr> <tr><td>6</td><td>account directory</td></tr> <tr><td>7</td><td>group directory</td></tr> <tr><td>8</td><td>not applicable</td></tr> <tr><td>9</td><td>byte stream</td></tr> <tr><td>10</td><td>hierarchical directory</td></tr> </table>	0	fixed	1	variable	2	undefined	3	spool block	4	root directory	5	not applicable	6	account directory	7	group directory	8	not applicable	9	byte stream	10	hierarchical directory
0	fixed																							
1	variable																							
2	undefined																							
3	spool block																							
4	root directory																							
5	not applicable																							
6	account directory																							
7	group directory																							
8	not applicable																							
9	byte stream																							
10	hierarchical directory																							
102	I32	<p>Dissallow file equation flag</p> <p>This item returns a value indicating whether or not file equations were disallowed when the file was opened. A value of zero indicates that file equations were allowed; a value of 1 indicates file equations were disallowed. This item is identical to bit 5:1 of the <b>foptions</b>.</p>																						

Table 4-6. FFILEINFO Itemnum/Item Values (continued)

Itemnum	Item Type	Item Description																
103	I32	<p>MPE Tape labels flag</p> <p>This item returns a value indicating whether or not a tape was opened labeled or unlabeled. A value of 0 indicates an unlabeled tape; a value of 1 indicates a labeled tape. This item is identical to bit 6:1 of the foptions.</p>																
104	I32	<p>File designator type</p> <p>This item returns the file designator type for this file. It is identical to bit 10:3 of the foptions. The following values can be returned by this item:</p> <table> <tr> <td>0</td> <td>A normal file open</td> </tr> <tr> <td>1</td> <td>\$STDLIST</td> </tr> <tr> <td>2</td> <td>\$NEWPASS</td> </tr> <tr> <td>3</td> <td>\$OLDPASS</td> </tr> <tr> <td>4</td> <td>\$STDIN</td> </tr> <tr> <td>5</td> <td>\$STDINX</td> </tr> <tr> <td>6</td> <td>\$NULL</td> </tr> <tr> <td>7</td> <td>\$STDERR</td> </tr> </table>	0	A normal file open	1	\$STDLIST	2	\$NEWPASS	3	\$OLDPASS	4	\$STDIN	5	\$STDINX	6	\$NULL	7	\$STDERR
0	A normal file open																	
1	\$STDLIST																	
2	\$NEWPASS																	
3	\$OLDPASS																	
4	\$STDIN																	
5	\$STDINX																	
6	\$NULL																	
7	\$STDERR																	
105	I32	<p>ASCII/Binary flag</p> <p>This item returns the setting of the file's ASCII/Binary flag. A value of 0 indicates that this is a binary file; a value of 1 indicates that this is an ASCII file. This item is identical to bit 13:1 of the foptions.</p>																
106	I32	<p>File domain</p> <p>This item returns the domain in which the file currently resides. It is identical to bit 14:2 of the foptions, and can return the following values:</p> <table> <tr> <td>0</td> <td>New file domain</td> </tr> <tr> <td>1</td> <td>Permanent file domain</td> </tr> <tr> <td>2</td> <td>Temporary file domain</td> </tr> </table>	0	New file domain	1	Permanent file domain	2	Temporary file domain										
0	New file domain																	
1	Permanent file domain																	
2	Temporary file domain																	

Table 4-7. 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
1083	GRAPH	GRAPH specification file
1084	SD	Self-describing file
1090	LOG	User logging log file

Table 4-7. FFILEINFO File Codes (continued)

Integer	Mnemonic	Description
1100	WDOC	Hewlett-Packard WORD document
1101	WDICT	Hewlett-Packard WORD hyphenation dictionary
1102	WCONF	Hewlett-Packard WORD configuration file
1103	W2601	Hewlett-Packard WORD 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	Hewlett-Packard Toolset root file
1141	TSD	Hewlett-Packard Toolset data file
1145	DRAW	Drawing file for Hewlett-Packard DRAW
1146	FIG	Figure file for Hewlett-Packard DRAW
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
1166	MDIST	Hewlett-Packard Desk distribution list
1167	MTEXT	Hewlett-Packard Desk text
1168	MARPA	ARPA messages file

**FFILEINFO****Table 4-7. FFILEINFO File Codes (continued)**

<b>Integer</b>	<b>Mnemonic</b>	<b>Description</b>
1169	MARPD	ARPA distribution list
1170	MCMND	Hewlett-Packard Desk abbreviated commands file
1171	MFRTM	Hewlett-Packard Desk diary free time list
1172	None	Reserved
1173	MEFT	Hewlett-Packard Desk external file transfer messages file
1174	MCRPT	Hewlett-Packard Desk encrypted item
1175	MSERL	Hewlett-Packard Desk 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
1236	BMAP	Base map specification file

Table 4-7. FFILEINFO File Codes (continued)

Integer	Mnemonic	Description
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	Hewlett-Packard WORD/150 document
1401	CWPTX	Reserved
1421	MAP	Hewlett-Packard MAP/3000 map specification file
1422	GAL	Reserved
1425	TTX	Reserved
1461	NMOBJ	Native mode object file
1462	PASLB	Pascal/XL source library

FOPEN FOPTIONS

Bits	(0:1)	(1:1)	(2:3)	(5:1)	(6:1)	(7:1)	(8:2)	(10:3)	(13:1)	(14:2)
Field	Reserved	Record Format Extension	File Type	Disallow :FILE	MPE Tape Labels	Carriage Control	Record Format	Default File Designator	ASCII BINARY	Domain
Meaning		0= don't use extended record format 1= use extended record format - only valid for (8:2 = 01 → byte stream)	00 0=STD 01 1=KSAM 01 0=RIO 01 1=KSAMXL 10 0=CIR 10 1=SPOOLFILE 11 0=MSG	0=Allow :FILE 1=No :FILE	0=Non-Labeled Tape 1=Labeled Tape	0= NOCCTL 1=CCTL	00=Fixed 01=Variable or Byte Stream 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\_002c

NOTE: Double lines indicate octal digit boundaries

Figure 4-3. Foption Bit Summary

# FFILEINFO

## 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 File	1= NOWAIT 0= 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	000= Read only 001= Write only 010= Write (Save) only 011= Append only 100= Read/Write 101= Update 110= Execute 001= Directory Read Access

LG200154\_003a

NOTE: Double lines indicate octal digit boundaries

Figure 4-4. Aoption Bit Summary

### Condition Codes

- CCE (2) Request granted.
- CCG (0) Not returned.
- CCL (1) Request denied. Access or calling sequence error.

### Related Information

- Intrinsics None
- Commands None
- Manuals *Getting System Information Programmer's Guide* (32650-90018)



---

## FFINDBYKEY

NM and CM callable.

Positions the record pointer at the beginning of the first record matching the key value comparison.

---

**Note** For KSAM files only.

---

### Syntax

I16V    CA    I16V    I16V    I16V FFINDBYKEY( <i>filenum,value,location,length,relop</i> );
---

### Parameters

*filenum*    **16-bit integer by value (required)**

Identifies the file number of the file to be positioned.

*value*    **character array (required)**

Contains a value that determines which record is read. This value is compared to the data contained in *location* in relation to the operator specified in *relop*.

*location*    **16-bit integer by value (required)**

Specifies the relative byte location in the record of the key being used. Bytes are numbered starting with 1. If *location*=0, the primary key is used.

*length*    **16-bit integer by value (required)**

Specifies the length of the key in bytes. If *length*=0, then the entire key is used. If *length* is less than the full key length (generic key), then only the length specified is used in the comparison with *relop*. The *length* parameter must be equal to or less than the full length of the key when the file was created. For numeric display keys or packed decimal keys, the full key length must be used.

## FFINDBYKEY

*relop*      **16-bit signed integer by value (required)**

Specifies the relational operator for the comparison of the key value of the file to the value specified in *value*. The record where the file is positioned has this relation to key value:

Value	Meaning
0	Equal
1	Greater than
2	Equal to or greater than

When *relop* is set to 1 or 2, the search is for an approximate key.

### Operation Notes

Split stack calls are permitted.

Can be used prior to a call to `FREADC` to position the chronological pointer to the record located by the specified key.

To locate and read a single record, use the `FREADBYKEY` intrinsic.

### Condition Codes

- CCE (2)      Request granted.
- CCG (0)      Request denied. The requested position was beyond the logical end-of-file or beginning-of-file.
- CCL (1)      Request denied. An error occurred: an I/O error occurred, the *relop* parameter could not be satisfied, a *length* less than the full length was specified for a key with numeric display or packed decimal format, or a key was not found when *relop*=0.

### Related Information

- Intrinsics      `FREADBYKEY`
- Commands      None
- Manuals      *Using KSAM XL* (32650-90168) and *KSAM/3000 Reference Manual* (30000-90079)

---

## FFINDN

NM and CM callable.

Positions the logical record pointer to the relative record number according to the key sequence.

---

**Note** For KSAM files only.

---

### Syntax

I16V    DV    I16V FFINDN( <i>filename,number,location</i> );
--

### Parameters

*filename*    **16-bit signed integer (required)**

Passes the file number of the file to be positioned.

*number*    **double by value (required)**

Specifies a record number relative to the first logical record in the file. Record numbers start with zero or one depending on the record numbering scheme specified at file creation. The lowest numbered record applies to the record with the lowest value in the specified key field. A negative record number positions the file pointer to the record with the smallest key value.

*location*    **16-bit signed integer by value (required)**

Passes the relative byte location in the record of the key to be used. The first byte of the record is considered 1. If *location*=0, the primary key is used.

### Operation Notes

Split stack calls are permitted.

When the relative record number is specified, be sure not to confuse this number with the chronological record number (the number of the record as it is stored in the file). The relative record number is based on the value of a specified key, not its location in a file.

(KSAM/3000) If **FFINDN** is used to position the pointer before calling another procedure that reads or updates the file in a shared environment, **FLOCK** must be called before calling **FFINDN**. After performing the read or update operation, unlock the file. If the file is locked after calling **FFINDN**, another user can change the pointer position without your program being aware of it.

## **FFINDN**

### **Condition Codes**

CCE (2)	Request granted.
CCG (0)	Request denied. The requested position was beyond the logical end-of-file.
CCL (1)	Request denied. An error occurred.

### **Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Using KSAM XL</i> (32650-90168) and <i>KSAM/3000 Reference Manual</i> (30000-90079)

---

## FGETINFO

NM and CM callable.

Returns access and status information about a file.

---

**Note** FGETINFO is provided for compatibility with MPE V/E-based systems only. It is recommended that FFILEINFO be used to access data.

---

### Syntax

I16V	CA	U16	U16						
FGETINFO( <i>filenum</i> , <i>formaldesig</i> , <i>foption</i> , <i>aoption</i>									
I16	I16	U16	U16	I16					
<i>lrecsize</i> , <i>devtype</i> , <i>ldevnum</i> , <i>hdaddr</i> , <i>filecode</i> ,									
I32	I32	I32	I32	I32	I16				
<i>lrecptr</i> , <i>eof</i> , <i>filelimit</i> , <i>logcount</i> , <i>physcount</i> , <i>blksize</i> ,									
U16	I16	I16	CA	I32					
<i>extsize</i> , <i>numextent</i> , <i>userlabels</i> , <i>creatorid</i> , <i>labaddr</i> );									

### Parameters

*filenum* **16-bit signed integer by value (required)**

Passes the file number of the file for which information is requested.

*formaldesig* **character array (optional)**

Returns the actual designator of the file being referenced, in the following format:

*filename.groupname.accountname*

The *formaldesig* array must be at least 28 bytes in length. When the actual designator is returned, unused bytes in the array are filled with blanks on the right. A nameless file returns an empty string.

The fully qualified name of the file referenced by *filenum* is returned as the value of this parameter. Only names which can be expressed using MPE-only semantics are returned. If the name of the object referenced by *filenum* can not be expressed using MPE-name semantics, a CCL condition code is returned. Calling FCHECK after this error will return FSERR391.

## FGETINFO

*foption*

### 16-bit unsigned integer by reference (optional)

Returns seven file access characteristics by setting bit groupings (refer to Figure 4-5) as follows:

Bits	Value/Meaning
14:2	Domain:  The file domain searched to locate the file:  00 New file 01 Old permanent file 10 Old temporary file 11 Old temporary or permanent file  The record format extension bit is returned as the <i>foption</i> (1:1) bit. Byte stream record format is represented as a record format extension of one with a variable record format ( <i>foption</i> (8:2) bits equal to 01.  Directories, symbolic links, device links, pipes and FIFO's cannot be represented by <i>foptions</i> . If the object referenced by <i>flenum</i>  (ASC) Not valid for asynchronous device files; however, these bits may have been set if file redirection was anticipated.
13:1	ASCII/Binary:  0 Binary 1 ASCII
10:3	Default file designator:  000 Actual file designator (same as formal file designator) 001 Actual file designator <b>\$STDLIST</b> 010 Actual file designator <b>\$NEWPASS</b> 011 Actual file designator <b>\$OLDPASS</b> 100 Actual file designator <b>\$STDIN</b> 101 Actual file designator <b>\$STDINX</b> 110 Actual file designator <b>\$NULL</b>

Bits	Value/Meaning
8:2	Record format <i>foption</i> : 00 Fixed-length records 01 Variable-length records 10 Undefined-length records 11 Spoolfile format records  (ASC) Not valid for asynchronous device files; however, “fixed-length” may have been set if file redirection was anticipated.
7:1	Carriage control: 0 No carriage-control character expected 1 Carriage-control character expected
6:1	MPE/iX tape label: 0 Nonlabeled tape 1 Labeled tape  (ASC) Not valid for asynchronous device files. May have been set if file redirection was anticipated.
5:1	Disallow <b>FILE</b> equation <i>foption</i> :  This option ignores any corresponding <b>FILE</b> command, so the specifications in the call to <b>FOPEN/HPFOPEN</b> take effect (unless overridden by those in the file label). The following bit settings apply:  1 Disallow <b>FILE</b> command 0 Allow <b>FILE</b> command

## FGETINFO

Bits	Value/Meaning
2:3	File type: 000 Standard file (STD) 001 KSAM/3000 file 010 Relative I/O file (RIO) 011 KSAM XL file 100 Circular file (CIR) 101 NM spoolfile 110 Message file (MSG)  (ASC) Set to 000 for asynchronous device files.
1:1	Record format extension: 0 Do not use extended record format 1 Use extended record format
0:1	Reserved for the operating system

*aoption*

### 16-bit unsigned integer by reference (optional)

Returns file access information by setting bit groupings (see Figure 4-6), in the following manner:

Bits	Value/Meaning
12:4	Access type <i>aoption</i> :  The type of access allowed users of this file: 0000 Read only 0001 Write only 0010 Write only (previous data not deleted) 0011 Append only 0100 Read/Write 0101 Update 0110 Execute 0111 Execute-Read 1000 Reserved 1001 Directory read access
11:1	Multirecord: 0 Nonmultirecord mode 1 Multirecord mode
10:1	Dynamic locking: 0 Disallow dynamic locking/unlocking 1 Allow dynamic locking/unlocking
8:2	Exclusive:  Specifies the access, from the time it is opened to the time it is closed: 00 Default 01 Exclusive access 10 Semi-Exclusive access 11 Shared access



Bits	Value/Meaning
7:1	Inhibit buffering:  Inhibits automatic buffering by the operating system and allows I/O to take place directly between the stack and the applicable hardware device:  0    Normal buffering 1    Inhibit buffering  (ASC) Not valid for asynchronous device files.
5:2	Multiaccess mode:  Provides the accessors with shared access to the file:  00   Nonmultiaccess 01   Intrajob multiaccess 10   Interjob multiaccess  (ASC) Not valid for asynchronous device files.
4:1	Nowait I/O:  Allows the accessors to initiate an I/O request and to have control returned before completion of the I/O.  0    Nowait I/O not in effect 1    Nowait I/O in effect
3:1	File copy access:  0    Access in file's native mode 1    Access as standard sequential file
0:3	Reserved for the operating system.

*lrecsize***16-bit signed integer by reference (optional)**

Returns the logical record size associated with the file:

- If the file was created as a binary file, this value is positive and expresses the size in half words.
- If the file was created as an ASCII file, this value is negative and expresses the size in bytes.

For interprocess communication (IPC), when there is a call to **FCONTROL** with *controlcode*=46, the value returned in *lrecsize* is the size of the data records, including the 4-byte header.

Maintained to ensure backward compatibility with MPE V/E-based systems. MPE V/E record size limits are imposed when **FGETINFO** returns record size information on all file types (STD, KSAM, RIO, CIR, MSG). If the record size of the specified file exceeds MPE V/E limits, a zero is returned. Use *itemnum* 67 of the **FFILEINFO** intrinsic or *itemnum* 30 of the **FLABELINFO** intrinsic to obtain valid MPE/iX record size data if the record size exceeds MPE V/E limits.

*devtype***16-bit signed integer by reference (optional)**

## FGETINFO

Returns the type and subtype of the device being used for a KSAM, RIO, circular, or message file, or devices such as a tape drive, printer, or terminal where bits (0:8) indicate device subtype, and bits (8:8) indicate device type. For standard disk files, bits (8:8)=00000011 and bits (0:8)=00001000 (indicate a 7933/35 disk drive).

If the file is not spooled or is opened as a spoolfile through the logical device, the returned *devtype* is actual. If an output file is spooled and was opened by device class name, *devtype* contains the type and subtype of the first device in its class. (This may be different from the device actually used.)

*ldevnum*      **16-bit unsigned integer by reference (optional)**

Returns the logical device number (ldev) associated with the device where the file label resides:

- If the file is a disk file, *ldevnum* is the location of the file label. (File data may reside on the same device as the file label.)
- If the file is spooled, *ldevnum* is a virtual device number that does not correspond to the system configuration I/O device list.
- If the file is located on a remote computer, linked by a DS point-to-point or X.25 link, the left eight bits (0:8) are the logical device number of the distributed system (DS) device.
- If the remote computer is linked by NS 3000/XL, the left eight bits (0:8) are the remote environment of the connection. The right eight bits (8:8) are the LDEV of the device on the remote computer where the file label resides.
- If the DS device for the RFA or the LDEV is 0, then *ldevnum* returns a 0.

*hdaddr*      **16-bit unsigned integer by reference (optional)**

Returns 2048. Maintained to provide backward compatibility with MPE V/E-based systems.

*filecode*    **16-bit signed integer by reference (optional)**

Returns the file code of a disk file.

(ASC) Not valid for asynchronous device files.

*lrecptr*      **32-bit signed integer by reference (optional)**

Returns the current logical record pointer setting. This value is the displacement in logical records from record number 0 in the file and identifies the record that would be accessed next by an FREAD or FWRITE call.

(ASC) Not valid for asynchronous device files.

*eof*          **32-bit signed integer by reference (optional)**

Returns the pointer setting of the last logical record currently in the file (equivalent to the number of logical records currently in the file). If the file does not reside on disk, this value is zero. For interprocess communication (IPC), when a call to FCONTROL with *itemnum*=46 is in effect, the number of records returned in *eof* includes open, close, and data records.

(ASC) Not valid for asynchronous device files.

<i>filelimit</i>	<b>32-bit signed integer by reference (optional)</b> Returns a number representing the last logical record that could exist in the file (the physical limits of the file). If the file does not reside on disk, this value is zero. (ASC) Not valid for asynchronous device files.
<i>logcount</i>	<b>32-bit signed integer by reference (optional)</b> Returns the total number of logical records passed to and from the program during the current file access.
<i>physcount</i>	<b>32-bit signed integer by reference (optional)</b> Returns the total number of physical I/O operations performed within the process, against the file, since the last FOPEN/HPFOPEN call.
<i>blksize</i>	<b>16-bit signed integer by reference (optional)</b> Returns the file block size: <ul style="list-style-type: none"> <li>■ If the file is binary, the value is positive and the size is in half words.</li> <li>■ If the file is ASCII, the value is negative and the size is in bytes.</li> </ul> Maintained for backward compatibility with MPE V/E-based systems. MPE V/E block size limits are imposed when FGETINFO returns block size information on all file types (STD, KSAM, RIO, CIR, MSG). If the block size of the specified file exceeds MPE V/E limits, 0 is returned. Use <i>itemnum</i> =68 of the FFILEINFO intrinsic or <i>itemnum</i> =31 of the FLABELINFO intrinsic to get valid MPE/iX block size data if the block size exceeds MPE V/E limits. (ASC) Not valid for asynchronous device files.
<i>extsize</i>	<b>16-bit unsigned integer by reference (optional)</b> Maintained to provide backward compatibility with MPE V/E-based systems. (ASC) Not valid for asynchronous device files.
<i>numextent</i>	<b>16-bit signed integer by reference (optional)</b> Maintained to provide backward compatibility with MPE V/E-based systems. (ASC) Not valid for asynchronous device files.
<i>userlabels</i>	<b>16-bit signed integer by reference (optional)</b> Returns the number of user labels defined for the file during creation. If the file is not a disk file, this number is zero. When an old file is opened for overwrite output, the value of <i>userlabels</i> is not reset, and old user labels are not destroyed. (ASC) Not valid for asynchronous device files.
<i>creatorid</i>	<b>character array (optional)</b> An unqualified form of the file owner's name is returned as the value of this parameter. The file owner is not necessarily the file's creator. File ownership may be changed using the :ALTFILE command.

## FGETINFO

A symbolic zero (ASCII 48 in decimal) is returned as the file owner for root directories, accounts, and MPE groups created prior to the POSIX release.

If the file is not located in the account in which the file owner is a member, a blank file owner name is returned.

(ASC) Not valid for asynchronous device files.

*labaddr*

**32-bit signed integer by reference (optional)**

Returns a zero. Maintained for backward compatibility with MPE V/E-based systems.

(ASC) Not valid for asynchronous device files.

### FOPEN FOPTIONS

Bits	(0:1)	(1:1)	(2:3)	(5:1)	(6:1)	(7:1)	(8:2)	(10:3)	(13:1)	(14:2)
Field	Reserved	Record Format Extension	File Type	Disallow :FILE	MPE Tape Labels	Carriage Control	Record Format	Default File Designator	ASCII BINARY	Domain
Meaning		0= don't use extended record format 1= use extended record format – only valid for (8:2 = 01 → byte stream)	00 0=STD 00 1=KSAM 01 0=RIO 01 1=KSAMXL 10 0=CIR 10 1=SPOOLFILE 11 0=MSG	0=Allow :FILE 1=No :FILE	0=Non-Labeled Tape 1=Labeled Tape	0= NOCCTL 1=CCTL	00=Fixed 01=Variable or Byte Stream 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\_002c

NOTE: Double lines indicate octal digit boundaries

**Figure 4-5. Foption Bit Summary**

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 File	1=NOWAIT 0= 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	000=Read only 001=Write only 010=Write (Save) only 011=Append only 100=Read/Write 101=Update 110=Execute 001=Directory Read Access

LG200154\_003a

NOTE: Double lines indicate octal digit boundaries

Figure 4-6. Aoption Bit Summary

**Operation Notes**

Returns access and status information about a file located on any device. The file must be opened by the calling process at the time of the FGETINFO call.

**Condition Codes**

- CCE (2) Request granted.
- CCG (0) Not returned.
- CCL (1) Request denied. An error occurred.

**Related Information**

- Intrinsics FFILEINFO, FOPEN
- Commands None
- Manuals *Getting System Information Programmer's Guide (32650-90018)*, *Using KSAM XL (32650-90168)*, and *KSAM/3000 Reference Manual (30000-90079)*

---

## FGETKEYINFO

NM and CM callable.

Requests access and status information about a KSAM file.

---

**Note** For KSAM files only.

---

### Syntax

I16V      BA      BA FGETKEYINFO( <i>filenum,param,control</i> )
---

### Parameters

*filenum*      **16-bit signed integer by value (required)**

Passes the file number of the file about which information is requested.

*param*      **byte array (required)**

Returns information describing the key information for a KSAM file. The length depends on the number of keys in the file. The file size is given as the number of sectors.

For KSAM XL files, refer to the *Using KSAM XL* (32650-90168) for a complete description of this parameter.

For KSAM/3000 files, refer to the *KSAM/3000 Reference Manual* (30000-90079) for a complete description of this parameter.

*control*      **byte array (required)**

Passes 128-words of control information about the key file.

### Operation Notes

When a KSAM file is opened, information can be requested about it using this intrinsic. The returned *param* parameter provides static information defined for the keys at the time it was created. The *control* parameter provides dynamic information about the use of the file from the time it was created. It counts the number of times the file was referenced by intrinsics, and the date and time it was created, closed, updated, or written to.

**Condition Codes**

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred; insufficient space was declared for <i>param</i> or <i>control</i> , an illegal file number was specified, or the DB register is not set to the user stack.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Using KSAM XL</i> (32650-90168) and <i>KSAM/3000 Reference Manual</i> (30000-90079)

---

## FINDJCW

NM and CM callable.

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

### Syntax

CA	U16	I16
FINDJCW( <i>jcwname</i> , <i>jcwvalue</i> , <i>jcwstatus</i> );		

### Parameters

*jcwname*      **character array (required)**

Passes the name of the job control word (JCW) to be found. This name can contain up to 255 alphanumeric characters, starting with a letter and delimited by a nonalphanumeric character, such as a blank.

*jcwvalue*      **16-bit unsigned integer by reference (required)**

Returns the value of *jcwname*, if *jcwname* is found. If *jcwname* is not found, *jcwvalue* is unchanged.

*jcwstatus*      **16-bit signed integer by reference (required)**

Returns the execution status of the intrinsic, as follows:

Value	Meaning
0	Successful
1	<i>Jcwname</i> > 255 characters
2	<i>Jcwname</i> does not start with an alpha character
3	<i>Jcwname</i> not found



## Operation Notes

There are three types of JCWs in the system:

- User-defined
- System-defined
- System-reserved

FINDJCW can return the value of any type of JCW. The system-reserved JCWs are:

- HPDATE
- HPDAY
- HPHOUR
- HPMINUTE
- HPMONTH
- HPYEAR

## Related Information

Intrinsics      GETJCW, PUTJCW, SETJCW

Commands      SETJCW, SHOWJCW

Manuals      *Interprocess Communication Programmer's Guide* (32650-90019)  
and *Command Interpreter Access and Variables Programmer's Guide*  
(32650-90011)

---

## FINTEXTIT

NM and CM callable.

Causes the return from your interrupt procedure.

### Syntax

U16V FINTEXTIT( <i>interruptstate</i> );
---

### Parameters

*interruptstate* **16-bit unsigned integer by value (optional)**

Passes the state in which software interrupts should be left. Bit (15:1) controls this as:

Value	Meaning
1	Enable software interrupts (Default)
0	Disable software interrupts

### Operation Notes

Software interrupts are set on return, according to *interruptstate*. If *interruptstate* is omitted, the software interrupts are enabled by default.

### Related Information

Intrinsics      None

Commands        None

Manuals        *Accessing Files Programmer's Guide* (32650-90017) and *Interprocess Communication Programmer's Guide* (32650-90019)

## FINTSTATE

NM and CM callable.

Enables/disables all software interrupts against the calling process.

### Syntax

U16	U16V
<i>oldstate</i> := FINTSTATE( <i>interruptstate</i> );	

### Functional Return

*oldstate*      **16-bit unsigned integer (assigned functional return)**  
 Returns the previous state (enabled or disabled) of the software interrupts.

### Parameters

*interrupt-state*      **16-bit unsigned integer by value (required)**  
 Passes a flag enabling/disabling software interrupts through bit (15:1):

Value	Meaning
0	Disable software interrupts
1	Enable software interrupts

### Operation Notes

The software interrupt facility enables **FREAD**/**FWRITE** completion processing with an interrupt procedure. A call to **FREAD** or **FWRITE** is necessary to initiate the I/O request. Both of these intrinsics return to the process as soon as the request has been started. When the operation completes, the program is interrupted and control goes to the chosen interrupt procedure. This performs whatever processing is necessary and then resumes the original program.

Software interrupts are armed for a particular file by specifying the interrupt procedure's *plabel* in an **FCONTROL** call with *itemnum*=48. If *aooption* (4:1) is set to 0, selecting *itemnum*=48 resets it to 1. Use **IOWAIT** or **IODONTWAIT** if *itemnum*=48 is used.

Software interrupts are inhibited just before entering an interrupt procedure. This is done to stop unwanted nesting of the interrupt procedures. Each interrupt procedure should call **FINTEXTIT** to reenable other interrupts just before it exits.

Software interrupts are automatically inhibited before a **(CTRL)Y** trap procedure. The trap procedure can allow software interrupts only by calling the **FINTSTATE** intrinsic. The **RESETCONTROL** intrinsic restores the interrupt state of the process to its pre-**(CTRL)Y** value (unless the trap procedure issues an **FINTSTATE** call; **RESETCONTROL** makes no change).

## FINTSTATE

The first parameter of the software interrupt procedure is the file number of the file that caused the interrupt.

It is necessary to issue a call to the `IODONTWAIT` intrinsic against the file in order to complete the request. When reading, the *buffer* parameter is ignored in the `FREAD` call. The data is moved to the array specified by the *buffer* parameter of `IODONTWAIT`.

An incomplete `FREAD/FWRITE` request can be aborted by issuing an `FCONTROL` call with an *itemnum* of 43 (abort nowait I/O).

Limitations:

- Only message (MSG) files allow software interrupts.
- No more than one incomplete `FREAD/FWRITE` can be outstanding for a particular file.
- The interrupt is held off while executing within the operating system, with the following exceptions:
  - `PAUSE` allows the interrupt.
  - `IOWAIT` allows the interrupt if the *flenum* parameter is set to 0.

In these instances, the intrinsic is reinvoked after the interrupt procedure executes.

- Do not use `FINTSTATE` with remote files.

## Related Information

Intrinsics      None

Commands        None

Manuals        *Accessing Files Programmer's Guide* (32650-90017) and *Interprocess Communication Programmer's Guide* (32650-90019)

## FLABELINFO

NM and CM callable.

Returns information from the file label of a disk file.

### Syntax

```

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

### Parameters

*formaldesig* **character array (required)**

Passes a formal file designator interpreted according to MPE-escaped semantics. The file name must be terminated by a nonalphanumeric character other than a period (.), a slash (/), a hyphen (-), and an underscore (\_). The file referred to by *formaldesig* can be either an MPE file (i.e., one that uses MPE syntax), or it can be an HFS syntax name beginning with a dot or a slash.

If *formaldesig* is an escaped pathname:

- you cannot reference remote files
- it cannot express a name equivalent to *filename:envid*
- you cannot use the *device* parameter (*device=node#*) to specify the remote location of a device

If *formaldesig* follows MPE syntax, the file name can include password, group, and account specifications. The file name can backreference a file equation and optionally be preceded by an asterisk.

The file referred to by *formaldesig* may reside either in an MPE group or in an HFS directory. For files located in HFS directories, traverse directory entries (TD) access is required to all directories specified in *formaldesig*. If there is no TD access, FLABELINFO fails and a file system error code (398) is returned in the *fserrorcode* parameter.

If the file can be named using both MPE syntax and HFS syntax (for example, FILEA.MYGROUP.MYACCT and /MYACCT/MYGROUP/FILEA), the file can be either permanent or temporary. If a temporary and a permanent file have the same name, FLABELINFO returns information about the temporary file only.

## FLABELINFO

*mode*      **16-bit signed integer by value (required)**

Passes an option specifying the valid backreferencing to file equations for the file.

Bits	Value/Meaning
0:11	Reserved for future use.
11:1	Symbolic Link Traversal
	0                      To traverse through symbolic links, if they exist.
	1                      Do not traversing through symbolic links, if they exist.
12:2	Caller Privilege Level
	Allows the caller to pretend to be less privileged. The privilege level is passed in this field.
14:2	File Equations
	0                      Use file equations if they exist.
	1                      A file equation must be used.
	2                      Do not use a file equation.

*fserrorcode*      **16-bit signed integer by reference (required)**

Returns a value indicating whether an error or warning occurred when FLABELINFO attempted to return requested information:

- A value of zero indicates that no errors were encountered.
- A positive value is a file system error code and indicates that an error was encountered and no information was returned in *item*.
- A -1 indicates that an item error or warning has occurred. Check the *itemerror* parameter to determine which item(s) has an error/warning and what it is.

*itemnum*      **16-bit signed integer array (required)**

Specifies which *item* value is to be returned. (Refer to Table 4-8.)

To indicate the end of the list, place a zero in the element following the last *itemnum*.

*item*            **record (required)**

Returns the value of the item specified in the corresponding *itemnum*. (Refer to Table 4-8.)

*Itemnum/items* are paired such that the *n*th field of the *item* record corresponds to the *n*th element of the *itemnum* array.

*itemerror*    **16-bit signed integer array (required)**

Returns an error number corresponding to the items specified in the *itemnum* array. The *itemnum/item* and *itemerror* parameters are paired such that the *n*th element of the *itemerror* array corresponds to the *n*th element of the *itemnum* array.

If a value in the *itemerror* array is negative, a warning exists for the corresponding item. If the value is positive, an error was detected for the corresponding item. The absolute value of each value is a file system error number.

**FLABELINFO**

**Table 4-8. FLABELINFO Itemnum/Item Values**

Itemnum	Mnemonic	Item Description
1	CA	<p>File name (8 bytes):</p> <p>The file name component for the file referenced in <i>formaldesig</i> is returned as the value. If the file name is not expressible using MPE-only semantics, a file system error code (391) is returned in the associated <i>itemerror</i>.</p>
2	CA	<p>Group name (8 bytes):</p> <p>The group name component for the file referenced in <i>formaldesig</i> is returned as the value. If the group name is not expressible using MPE-only semantics, a file system error code (391) is returned in the associated <i>itemerror</i>.</p>
3	CA	<p>Account name (8 bytes):</p> <p>The account name component for the file referenced in <i>formaldesig</i> is returned as the value. If the account name is not expressible using MPE-only semantics, a file system error code (391) is returned in the associated <i>itemerror</i>.</p>
4	CA	<p>File creator name (8 bytes):</p> <p>An unqualified form of the file owner's name is returned as the value. The file owner is not necessarily the file's creator.</p> <p>A symbolic zero (ASCII 48 in decimal) is returned as the file owner for root directories, MPE accounts, and MPE groups created prior to release 4.5.</p> <p>If the file is not located in the account where the file owner is a member, a blank file owner name is returned. Use <i>itemnum</i>=43 to obtain the full file owner name.</p>
5	U32	<p>Security matrix for access:</p> <p>Returns the file's security matrix. This value does not indicate the actual security enforced for a file, since group and account security masks can also restrict access. This field is ignored if an ACD is active on a file.</p>
6	U16	<p>File creation date:</p> <p>The date in <b>CALENDAR</b> intrinsic format. Either creator (C) or manager (AM if file is within account, otherwise SM) access required.</p> <p>Zero is returned as the creation date for root directories, MPE accounts, and MPE groups created prior to release 4.5.</p>
7	U16	<p>Last access date:</p> <p>The date in <b>CALENDAR</b> intrinsic format. May not be up-to-date when the file is open.</p> <p>Zero is returned as the last access date for root directories, MPE accounts, and MPE groups created prior to release 4.5.</p>
8	U16	<p>Last modification date:</p> <p>The date in <b>CALENDAR</b> intrinsic format. May not be up-to-date when the file is open.</p> <p>Zero is returned as the modification date for root directories, MPE accounts, and MPE groups created prior to release 4.5.</p>



Table 4-8. FLABELINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
9	I16	File code of disk file
10	U16	Number of user labels written: May not be up-to-date when the file is open.
11	U16	Number of user labels available: May not be up-to-date when the file is open.
12	I32	Total number of logical records possible in the file: Equivalent to the file limit measured in logical records.
13	U16	File options: The record format extension bit is returned as the <i>foption</i> (1:1) bit. Byte stream record format is represented as a record format extension of one with a variable record format ( <i>foption</i> (8:2) bits equal to 01).  Directories, symbolic links, device links, pipes and FIFO's cannot be represented by <i>foption</i> . If the object referenced by <b>filenum</b> is is an object, MPE error 399 is returned in the associated <i>itemerror</i> .  Refer to the <i>foption</i> figure.
14	I16	Record size: Maintained for compatibility with MPE V/E-based systems. (If a zero is returned, use <i>itemnum</i> 30 instead.)
15	I16	Block size: Maintained for compatibility with MPE V/E-based systems. (If a zero is returned, use <i>itemnum</i> 31 instead.)
16	I16	Maximum number of extents: Maintained for compatibility with MPE V/E-based systems. (If a zero is returned, use <i>itemnum</i> 32 instead.)
17	I16	Last extent size: Indicates sectors. May not be up-to-date when the file is open.
18	I16	Extent size: Indicates sectors. (If a zero is returned, use <i>itemnum</i> 32 instead.)

**FLABELINFO**

**Table 4-8. FLABELINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
19	U32	Number of logical records in file: Equivalent to EOF. May not be up-to-date when the file is open.
20	U32	File allocation time: The time when file was last restored (in <b>CLOCK</b> intrinsic format). Zero is returned as the file allocation time for root directories, MPE accounts, and MPE groups created prior to release 4.5.
21	U16	File allocation date: The date when the file was last restored (in <b>CALENDAR</b> intrinsic format). Zero is returned as the file allocation date for root directories, MPE accounts, and MPE groups created prior to release 4.5.
22	I32	Number of open/close records: MSG files only. May not be up-to-date when the file is open.
23	CA	Device name (8 bytes)
24	U32	Last modification time: The time when the file was last modified (in <b>CALENDAR</b> intrinsic format). May not be up-to-date when the file is open.
25	CA	First user label (user label 0) (256 bytes): May not be up-to-date when the file is open. For all but PM files, Manager capability (AM if file is within account, otherwise SM) or read/write (R/W) access required. For PM files, Manager capability is required; otherwise, <b>FLABELINFO</b> fails with <b>FSERR93</b> .
27	REC	Unique file identifier (UFID) (20 bytes)
28	U32	Total number of bytes allowed in file: Equivalent to the file limit measured in bytes. May not be up-to-date when the file is open.
29	U32	Start of file offset: Indicates the byte offset where user data starts.
30	U32	Record size (indicates bytes)
31	U32	Block size (indicates bytes)
32	U32	Extent size (indicates bytes)
33	CA	File lockword (8 bytes): Returned if you are the file creator, account manager, or system manager.

Table 4-8. FLABELINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
34	CA	<p>Volume restriction (34 bytes):</p> <p>The last two characters indicate the type of restriction, as follows:</p> <ul style="list-style-type: none"> <li>0 File is placed on the specified volume at creation</li> <li>1 File can be placed on any volume containing the specified class at creation</li> <li>2 File can be placed on any volume within the specified volume set at creation (Default)</li> </ul>
35	CA	<p>Volume set names (32 bytes):</p> <p>No restrictions.</p>
36	U32	<p>Transaction management log set id (4 bytes)</p> <p>No restrictions.</p>
37	U16	<p>Logical device number</p>
38	REC	<p>Terminated HFS-syntax system absolute pathname:</p> <p>Upon input, the first four bytes are interpreted as a 32-bit unsigned integer specifying the maximum available buffer size in bytes. This maximum available buffer size does not include the four bytes used to represent this size. Upon output, the first four bytes represent the pathname length excluding the null terminator as a 32-bit unsigned integer. The pathname is returned in bytes following the pathname length. Bytes beyond the pathname terminator are undefined. If the maximum available buffer size is incorrect upon input, variables allocated near the buffer can be overwritten or a bounds violation could occur. A zero pathname length is returned for unnamed new files and when an error occurs. Zero is the minimum buffer length upon input for this <i>itemnum</i>.</p> <p style="text-align: center;">Format of the buffer on input:</p> <pre> byte      #1  #2  #3  #4   #1  ...  #N +-----+-----+-----+-----+-----+ ... +-----+             maximum                                     length = N                     +-----+-----+-----+-----+ ... +-----+ </pre> <p style="text-align: center;">Format of the buffer on output:</p> <pre> byte      #1  #2  #3  #4   #1  ...  #L  ...  #N +-----+-----+-----+-----+-----+ ... +-----+ ... +-----+             length =             /               \0                           (L - 1)   +-----+-----+-----+-----+-----+ ... +-----+ ... +-----+ </pre>

**FLABELINFO**

**Table 4-8. FLABELINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
39	U32	The current number of hard links to the file
40	I32	Time of last file access (clock format): The bit assignments are: bits 0-7 = hours bits 8-15 = minutes bits 16-23 = seconds bits 24-31 = tenths of seconds
41	I32	Time of last file status change (clock format): The bit assignments are: bits 0-7 = hours bits 8-15 = minutes bits 16-23 = seconds bits 24-31 = tenths of seconds
42	U16	Date of the last file status change (calendar format): The bit assignments are: bits 0-7 = year of century bits 8-15 = day of the year
43	CA	File owner (32 bytes): The full file owner name. Unused characters are filled with blanks. A symbolic zero (ASCII 48 in decimal) is returned as the file owner for root directories, accounts, and MPE groups created prior to release 4.5.
44	I32	File owner identifier: The file owner identifier (UID). Zero is returned as the file owner ID for root directories, MPE accounts, and MPE groups created prior to release 4.5.
45	CA	File group (32 bytes): The file group name. Unused characters are filled with blanks. A symbolic zero (ASCII 48 in decimal) is returned as the group for root directories where GIDs have not been explicitly assigned.
46	I32	File group identifier: The file group identifier (GID). Zero is returned as the group ID for root directories where GIDs have not been explicitly assigned.

Table 4-8. FLABELINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
47	U32	<p>File type:</p> <p>Following are valid file types that can be returned:</p> <ul style="list-style-type: none"> <li>0 = Ordinary file</li> <li>1 = KSAM/3000</li> <li>2 = RIO</li> <li>3 = KSAM XL</li> <li>4 = CIR</li> <li>5 = Native Mode Spool File</li> <li>6 = MSG</li> <li>7 = N/A</li> <li>8 = N/A</li> <li>9 = Directory</li> <li>10-11= N/A</li> <li>12 = Pipe</li> <li>13 = FIFO</li> <li>14 = Symbolic Link</li> <li>15 = Device Link</li> </ul>
48	U32	<p>Record type:</p> <p>Following are valid record types that can be returned:</p> <ul style="list-style-type: none"> <li>0 = fixed</li> <li>1 = variable</li> <li>2 = undefined</li> <li>3 = spool block</li> <li>4 = root directory</li> <li>5 = N/A</li> <li>6 = account directory</li> <li>7 = group directory</li> <li>8 = N/A</li> <li>9 = byte stream</li> <li>10 = hierarchical directory</li> </ul>
49	I64	<p>Current file size (in bytes):</p> <p>The value returned represents the current position of the end-of-file (EOF) and may not reflect the number of bytes actually occupied by the file on disk if the file is sparsely allocated.</p>

**FLABELINFO**

**Table 4-8. FLABELINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
50	I32	<p>KSAM XL File Version:</p> <p>This item returns a value indicating the version number of a KSAM XL file. A value of 1 indicates an original type KSAM XL file. A value of 2 indicates the next generation KSAM XL file. A value of zero is returned if the file is not a KSAM XL file.</p>
51	I32	<p>KSAM XL Parameters:</p> <p>This item returns the <i>ksamparam</i> for KSAM XL files, which is an array that specifies the characteristics for up to 16 keys for the file.</p>
52	I32	<p>MPE/iX Device Type:</p> <p>This item returns the following values for the following types of devices:</p> <ul style="list-style-type: none"> <li>0=Disk device</li> <li>1=Tape device</li> <li>2=Terminal device</li> <li>3=Printer device</li> <li>4=Remote device</li> <li>5=Ports device</li> <li>6=Reserved</li> <li>7=Streams device</li> <li>8=Sockets device</li> </ul>
53	I16	<p>Secure/Release:</p> <p>This item returns a value indicating whether the file is currently secured or released. A value of 1 indicates that the file is secured. A value of zero indicates that the file is released.</p>
54	I32	<p>Setuid flag</p> <p>This item returns a value indicating whether or not the setuid flag is on for the specified file. When the setuid flag is on for a program file (the value is 1), the program will execute under the identity of the file's owner rather than assuming the identity of the process that invoked the program. A value of zero indicates the flag is off.</p>
55	I32	<p>Setgid flag</p> <p>This item returns a value indicating whether or not the setgid flag is on for the specified file. When the setgid flag is on for a program file (the value is 1), the program will execute with the GID membership of the file rather than assuming the GID membership of the invoking process. A value of zero indicates that the flag is off.</p>

FOPEN FOPTIONS

Bits	(0:1)	(1:1)	(2:3)	(5:1)	(6:1)	(7:1)	(8:2)	(10:3)	(13:1)	(14:2)
Field	Reserved	Record Format Extension	File Type	Disallow :FILE	MPE Tape Labels	Carriage Control	Record Format	Default File Designator	ASCII BINARY	Domain
Meaning		0= don't use extended record format 1= use extended record format – only valid for (8:2 = 01 → byte stream)	00 0=STD 00 1=KSAM 01 0=RIO 01 1=KSAMXL 10 0=CIR 10 1=SPOOLFILE 11 0=MSG	0=Allow :FILE 1=No :FILE	0=Non-Labeled Tape 1=Labeled Tape	0= NOCCTL 1=CCTL	00=Fixed 01=Variable or Byte Stream 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\_002c

NOTE: Double lines indicate octal digit boundaries

Figure 4-7. Foption Bit Summary

Condition Codes

- CCE (2) Request granted.
- CCG (0) Not returned. U\CCL (1)\Request denied. An error occurred. Refer to the *ferrorcode* and *itemerror* parameters for more information.

Related Information

- Intrinsics None
- Commands None
- Manuals *Getting System Information Programmer's Guide* (32650-90018)

---

## FLOCK

NM and CM callable.

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

### Syntax

I16V    U16V FLOCK( <i>filename</i> , <i>lockflag</i> );
---

### Parameters

*filename*    **16-bit signed integer by value (required)**

Passes the file number of the file whose global resource identification number (RIN) is to be locked.

*lockflag*    **16-bit unsigned integer by value (required)**

Specify either conditional or unconditional locking by setting bit (15:1) as follows:

Value	Meaning
0	Locking takes place only if the flock_semaphore of the global unique file descriptor (gufd) is used to control access to the file. If the RIN is locked, control returns immediately to the calling process, with condition code CCG (0).
1	Locking takes place unconditionally. If the file cannot be locked immediately, the calling process suspends until the file can be locked.

### Condition Codes

The following condition codes are possible when *lockflag* bit (15:1)=1:

CCE (2)    Request granted.

CCG (0)    Not returned.

CCL (1)    Request denied. This file was not opened with the dynamic locking *aooption* bit (10:1) specified in the FOPEN/HPFOPEN intrinsic, or there was an attempt to lock more than one RIN and the calling process does not have multiple RIN (MR) capability.



The following condition codes are possible when *lockflag* bit (15:1)=0:

- |         |  |
|---------|--|
| CCE (2) | Request granted.   |
| CCG (0) | Request denied. The file was locked by another process.  |
| CCL (1) | Request denied. This file was not opened with the dynamic locking <i>aoption</i> bit (10:1) specified in the <b>FOPEN/HPFOPEN</b> intrinsic, or there was an attempt to lock more than one RIN and the calling process does not have multiple RIN (MR) capability. |

### Operation Notes

On MPE/iX when a file is created (opened as a new file) with dynamic locking only one accessor (the creator) is allowed. A subsequent call to **FLOCK** against such a file (although unnecessary) will succeed because the dynamic locking bit is not set to zero. On MPE V, when a new file is created with dynamic locking the the dynamic locking bit is set to zero. For this reason, a subsequent call to **FLOCK** will fail on MPE V.

Accessors that have opened a file with the dynamic locking option enabled must access the file through the **FLOCK** intrinsic to gain exclusive access to the file. Since the use of this intrinsic is discretionary, however, all accessors must agree to use **FLOCK** when writing to a file to guarantee exclusive access. File locking is advised, but is not mandated by MPE/iX.

### Related Information

- |            |   |
|------------|---|
| Intrinsics | None  |
| Commands   | None  |
| Manuals    | <i>Resource Management Programmer's Guide</i> (32650-90024) |

---

## 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.

### Syntax

I32      I16
FLUSHLOG( <i>index</i> , <i>logstatus</i> );

### Parameters

*index*      **32-bit signed integer by reference (required)**

Passes the access to the user logging system. This is the value returned in the *index* parameter of the OPENLOG intrinsic.

*logstatus*      **16-bit signed integer by reference (required)**

Returns one of the following values, indicating the success/failure of the intrinsic call:

Value	Meaning
0	Successful
4	Incorrect <i>index</i> parameter passed
7	Illegal capability; user logging (LG) or system supervisor (OP) capability required
9	Error occurred while writing
12	System is out of disk space, user logging cannot proceed
14	Invalid access
15	End-of-file encountered

### Operation Notes

The FLUSHLOG intrinsic writes the contents of the user logging memory buffer to the disk destination file. This helps to preserve the contents of the memory buffer in the event of a system failure. This intrinsic does not write special records.

### Related Information

Intrinsics      None

Commands      None

Manuals      *User Logging Programmer's Guide (32650-90027)*

## 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

### Syntax

U16V    CA FMTCALENDAR( <i>date</i> , <i>formatdate</i> );
---

### Parameters

*date*            **16-bit unsigned integer by value (required)**

Passes the calendar date, in the same format as the CALENDAR intrinsic:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*formatdate*    **character array (required)**

Returns the formatted calendar date in a 17-character array. If the day of the month is less than 10, a blank precedes it, for example,

FRI, JAN 6, 1989

### Operation Notes

If invalid input values are supplied, the output values returned are unpredictable; an error status is not returned.

### Related Information

Intrinsics	None
Commands	None
Manuals	None

---

## 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

### Syntax

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

### Parameters

*time*            **32-bit signed integer by value (required)**

Returns the time of day in the same format as the `CLOCK` intrinsic:

Bits	Value/Meaning
24:8	Tenths of seconds
16:8	Seconds
8:8	Minute of hour
0:8	Hour of day

*formattime*    **character array (required)**

Returns the formatted time of day in an 8-character array. If the hour is a single digit (prior to 10:00), the array contains a leading blank as follows:

7:39 AM

### Operation Notes

If invalid input values are supplied, the output values returned are unpredictable; an error status is not returned.

### Related Information

Intrinsics	None
Commands	None
Manuals	None

## 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
```

### Syntax

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

### Parameters

*date*            **16-bit unsigned integer by value (required)**

Passes the date in the same format as the CALENDAR intrinsic:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*time*            **32-bit signed integer by value (required)**

Passes the time in the same format as the CLOCK intrinsic:

Bits	Value/Meaning
24:8	Tenths of seconds
16:8	Seconds
8:8	Minute of hour
0:8	Hour of day

*datetime*       **character array (required)**

Returns the formatted date and time in a 27-character array. If the day of the month is less than 10 or the time is one digit (before 10:00), a blank precedes it. For example,

```
FRI, JAN 6, 1989, 7:39 AM
```

## **FMTDATE**

### **Operation Notes**

If invalid input values are supplied, the output values returned are unpredictable; an error status is not returned.

### **Related Information**

Intrinsics	None
Commands	None
Manuals	None

## FOPEN

NM and CM callable.

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

### Syntax

```

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

```

### Functional Return

*filenum*      **16-bit signed integer (assigned functional return)**

Returns a unique file number identifying the opened file.

### Parameters

*formaldesig*      **character array (optional)**

Passes a formal file designator interpreted according to MPE-escaped semantics. The file name must be terminated by a nonalphanumeric character other than a period (`.`), a slash (`/`), a hyphen (`-`), and an underscore (`_`).

The file referred to by *formaldesig* can be either an MPE file (i.e., one that uses MPE syntax) or it can follow HFS syntax. If *formaldesig* follows MPE syntax, the file name can include password, group, and account specifications. The file name can backreference a file equation and optionally be preceded by an asterisk. If *formaldesig* follows HFS syntax, the file name must start with either a dot (`.`) or a slash (`/`).

The file referred to by *formaldesig* may reside either in an MPE group or in an HFS directory. For files located in HFS directories, traverse directory entries (TD) access is required to all directories specified in *formaldesig*. If there is no TD access, **FOPEN** fails and a call to **FCHECK** with *filenum* set to 0 will return a system error code (398) in the *fserrorcode* parameter.

If *formaldesig* is an escaped pathname:

- you cannot reference remote files
- it cannot express a name equivalent to *filename:envid*
- you cannot use the *device* parameter (*device=node#*) to specify the remote location of a device

## FOPEN

If *formaldesig* is the name of a user-defined file, it can begin with an asterisk (\*). If *formaldesig* is the name of a system-defined file, it can begin with a dollar sign (\$). When creating a KSAM file, *formaldesig* must be a unique file name, that is, one not currently existing in the permanent file directory.

The formal file designator can contain command interpreter variables and expressions that are evaluated before *formaldesig* is parsed and validated.

As the default, FOPEN creates a nameless file that can be read or written to, but not saved. (The domain option of a nameless file must specify a new file unless it is a device file.)

(ASC) This parameter is recommended for use with asynchronous device files because of the flexibility it provides.

*foption*

### 16-bit unsigned integer by value (optional)

Specifies up to eight different file characteristics by setting corresponding bit groupings:

---

#### Note

For old files, default conditions are specified in the file label.

Device characteristics may override some *foptions*.

---

Bits	Value/Meaning
------	---------------

14:2	Domain
------	--------

Indicates which file domain is searched to locate a file. A nameless disk file must always be a new file. A device file (such as a tape or terminal) always resides in the system file domain (permanent file directory). Always specify a device file as old or permanent.

The following bit settings are valid:

- 00 The file is new. No search is necessary.
- 01 The file is a permanent file. The system file domain (permanent file directory) is searched.
- 10 The file is a temporary file. The job file domain (temporary file directory) is searched.
- 11 The file is an old (permanent or temporary) file. The job file domain (temporary file directory) is searched. If not found, the system file domain is searched.

Default: 00



Bits	Value/Meaning
13:1	<p>ASCII/binary</p> <p>Indicates which code, ASCII or binary, a new file is in when written to a device that supports both codes. For disk files, this also affects padding (zeros for a binary file, blank characters for an ASCII file) that can occur when you issue a direct-write intrinsic call (<b>FWRITEDIR</b>) to a record that lies beyond the current logical end-of-file indicator. By default, magnetic tape and files are treated as ASCII files. This option is applicable only at file creation.</p> <p>The following bit settings are valid:</p> <p>0    Binary file</p> <p>1    ASCII file</p> <p>(ASC) Not valid for asynchronous device files.</p> <p>Default: 0</p>
10:3	<p>Designator</p> <p>Passes a value indicating a special file opening. Any of the following special files can be specified with the <i>formaldesig</i> parameter. For example, a file name of <b>\$STDLIST</b> opens standard list.</p> <p>The following bit settings are valid:</p> <p>000 The actual file designator is the same as the formal file designator.</p> <p>001 The actual file designator is <b>\$STDLIST</b>.</p> <p>010 The actual file designator is <b>\$NEWPASS</b>.</p> <p>011 The actual file designator is <b>\$OLDPASS</b>.</p> <p>100 The actual file designator is <b>\$STDIN</b>.</p> <p>101 The actual file designator is <b>\$STDINX</b>.</p> <p>110 The actual file designator is <b>\$NULL</b>.</p> <p>Default: 000</p> <p>For example, if you pass <b>MYFILE</b> in the <i>formaldesig</i> parameter, then using the designator option to equate it with <b>\$STDIN</b> is equivalent to allowing the file equation <b>FILE MYFILE=\$STDIN</b>.</p>

## FOPEN

Bits	Value/Meaning
	<p>The designator option is not equated with the <i>formal<del>desig</del></i> parameter if both of the following conditions are true:</p> <ul style="list-style-type: none"><li>■ The disallow file equation option <i>foption</i> bit (5:1) allows file equations for this file opening.</li><li>■ An explicit or implicit <b>FILE</b> command equating the formal file designator to a different actual file designator occurs in the job/session.</li></ul> <p>A leading * in a formal file designator passed by the <i>formal<del>desig</del></i> parameter overrides the disallow file equation option setting.</p>
8:2	<p>Record format</p> <p>Bit settings indicate internal record structure for a file. This option is applicable only at file creation.</p> <p>The following bit settings are valid:</p> <p>00 Fixed-length records. The file contains logical records of uniform length.</p> <p>Fixed-length records are supported by disk and magnetic tape devices only.</p> <p>01 Variable-length records. The file contains logical records of varying length. This format is restricted to records that are written in sequential order. The size of each record is recorded internally. The actual physical record size is determined by multiplying the record size (specified or default) plus one by the blocking factor, and adding one word for the end-of-block indicator.</p> <p>This format is the only valid combination with byte stream (1) record format extension.</p> <p>For new files, this option is not allowed when <b>NOBUF</b> is specified. If <b>NOBUF</b> is specified, then reads/writes are performed on the entire block, not just the record.</p> <p>Variable-length records are supported by disk and magnetic tape devices only.</p>

Bits	Value/Meaning
10	<p>Undefined-length records. The file contains records of varying length that were not written as variable-length files. By default, all files not on disk or magnetic tape are treated as containing undefined-length records. The file system makes no assumption about the amount of data that is useful. You must determine how much data is required. For undefined-length records, only the data supplied is written, with no information about its length.</p> <p>Undefined-length records are supported by all devices.</p> <p>Default: 00</p>
7:1	<p>Carriage control</p> <p>Indicates whether or not a carriage control directive is supplied in the calling sequence of each <b>FWRITE</b> call that writes records onto the file. This option is applicable only at file creation.</p> <p>The following bit settings are valid:</p> <p>0 No carriage-control directive is expected.</p> <p>1 Carriage-control directives are expected.</p> <p>Default: 0</p> <p>Carriage control is supplied only for ASCII files. This option and the ASCII/binary option (<i>foption</i> bit (13:1)) are mutually exclusive, and attempts to open new files with both binary and carriage control directives result in an access violation.</p> <p>A carriage-control character passed through the <i>controlcode</i> parameter of <b>FWRITE</b> is acted upon for files that have carriage-control specified in <b>HPFOPEN/FOPEN</b>. Embedded control characters are treated as data for files specified no carriage-control, and do not invoke spacing. Spacing action can be specified on files specified with carriage-control either by embedding the control in the record or by sending the control code directly through the <i>controlcode</i> parameter of <b>FWRITE</b>.</p>

## FOPEN

Bits	Value/Meaning
	<p>If a carriage-control character is sent to a file where the control cannot be executed directly (for example, line-spacing characters sent to a disk or tape file), the control character is embedded as the first byte of the record. If a carriage-control character is sent to other types of files, the control is transmitted to the driver.</p> <p>Control codes %400 through %403 are remapped to %100 through %103, so that they fit into one byte and thus can be embedded. Records written to the line printer with control codes %400 through %403 should contain only control information. A record written with control codes %400 through %403 and no data (<code>count=0</code>, or embedded control and <code>count=1</code>) does not cause physical I/O of any sort.</p> <p>To compute record size, the file system assumes carriage-control information to be part of the data record. Therefore, specifying the carriage-control option adds one byte to the record size when the file is originally created. For example, a specification of <code>REC=-132,1,F,ASCII;CCTL</code> results in a <i>recsize</i> of 133 bytes.</p> <p>In general, the entire record can be read (the size of which is returned in <i>itemnum</i>=67 of the <code>FFILEINFO</code> intrinsic). However, on writes to files where carriage-control characters are specified, the data transferred is limited to <i>recsize</i>-1 unless a control of 1 is passed, indicating that the data record is prefixed with embedded carriage-control characters.</p>
6:1	<p>Labeled tape</p> <p>0 No labeled tapes</p> <p>1 Labeled tapes</p> <p>(ASC) Not valid for asynchronous device files, but may be set if file direction is anticipated.</p> <p>Default: 0</p>

Bits	Value/Meaning
5:1	<p>Disallow file equation option</p> <p>Indicates whether or not to allow file equations. A leading * in a formal file designator can override the setting to disallow <b>FILE</b>. The following bit settings are valid:</p> <p>0 Allow <b>FILE</b> equations to override programmatic or system-defined file specifications.</p> <p>1 Disallow <b>FILE</b> equations from overriding programmatic or system-defined file specifications.</p> <p>Default: 0</p>
2:3	<p>File type option</p> <p>Indicates internal record structure used to access records in a file. If the file is old, this option is ignored. Specifying a designator option (<i>foption</i> bits (10:3)) value other than zero overrides this option. This option is applicable only at file creation.</p> <p>The following bit settings are valid:</p> <p>000 Standard (STD) file</p> <p>001 KSAM/3000 file</p> <p>010 Relative I/O (RIO) file</p> <p>011 KSAM XL file</p> <p>100 Circular (CIR) file</p> <p>101 NM spoolfile</p> <p>110 Message (MSG) file</p> <p>Default: 000</p>
1:1	<p>Record format extension</p> <p>Byte stream record format is specified by setting the record format, <i>foption</i> (8:2), to variable-length records (01) and the record format extension, <i>foption</i> (1:1), to byte stream record format (1). Zero is the default value for this option. Using any record format value other than variable-length records with the record format extension results in an <b>FSERR 49</b> (unimplemented function). Byte stream record format may only be specified for standard disk files. Specifying byte stream record format for any other type of file result in <b>FSERR 49</b> error.</p>

## FOPEN

Bits	Value/Meaning										
1:1 cont'd	Record format extension  Files created using byte stream record format are assigned file attributes which override values specified by <b>FOPEN</b> parameters. The file attributes are as follows: <table><thead><tr><th>Option</th><th>Description\Value</th></tr></thead><tbody><tr><td><i>foption</i> (13:1)</td><td>ASCII/binary\ASCII (1)</td></tr><tr><td><i>foption</i> (7:1)</td><td>Carriage control\NOCCCTL (0)</td></tr><tr><td><i>recsize</i></td><td>Logical record size\1 Byte (-1)</td></tr><tr><td><i>blockfactor</i></td><td>Blockfactor\1 Record/block</td></tr></tbody></table>	Option	Description\Value	<i>foption</i> (13:1)	ASCII/binary\ASCII (1)	<i>foption</i> (7:1)	Carriage control\NOCCCTL (0)	<i>recsize</i>	Logical record size\1 Byte (-1)	<i>blockfactor</i>	Blockfactor\1 Record/block
Option	Description\Value										
<i>foption</i> (13:1)	ASCII/binary\ASCII (1)										
<i>foption</i> (7:1)	Carriage control\NOCCCTL (0)										
<i>recsize</i>	Logical record size\1 Byte (-1)										
<i>blockfactor</i>	Blockfactor\1 Record/block										
0:1	Reserved for the operating system										

### *aoption* 16-bit unsigned integer by value (optional)

Specifies up to eight different file access options by setting corresponding bit groupings:

Bits	Value/Meaning
12:4	Access type  Indicates the type of access intended for the file. This option restricts/allows minimal use of file system intrinsics.  The following bit settings are valid:  0000 Allows read access only, if the file's security provisions specify read access. <b>FWRITE</b> , <b>FUPDATE</b> , and <b>FWRITEDIR</b> intrinsic calls cannot reference this file. The end-of-file (EOF) is not changed; the record pointer starts at 0.  0001 Allows write access only, if the file's security provisions allow write access. Any data written in the file prior to the current <b>FOPEN</b> request is deleted. <b>FREAD</b> , <b>FREADSEEK</b> , <b>FUPDATE</b> , and <b>FREADDIR</b> intrinsic calls cannot reference this file. The EOF is set to 0; the record pointer starts at 0. On magnetic tape an EOF is written when the file is closed even if no data is written.  0010 Allows write-save access only, if the file's security provisions allow write access. Previous data in the file is not deleted. <b>FREAD</b> , <b>FREADSEEK</b> , <b>FUPDATE</b> , and <b>FREADDIR</b> intrinsic calls cannot reference this file. The EOF is not changed; the record pointer starts at 0. Therefore, data is overwritten if a call to <b>FWRITE</b> is made. The system changes this value to append for message files.

Bits	Value/Meaning
0011	Allows append access only if the file's security provisions allow either append or write access. <b>FREAD</b> , <b>FREADDIR</b> , <b>FREADSEEK</b> , <b>FUPDATE</b> , <b>FSPACE</b> , <b>FPOINT</b> , and <b>FWRITEDIR</b> intrinsic calls cannot reference this file. Data written by the <b>FWRITE</b> intrinsic is appended to the EOF, thereby extending the EOF. When a file is opened for append access, it is impossible to overwrite data in the file. For disk files, the EOF is updated after each <b>FWRITE</b> call. Therefore, data cannot be overwritten.
0100	Allows read/write (I/O) access only if the file's security provisions allows both read and write access. If both read and write access are not allowed, the access type specified in the security provisions (either read or write) is allowed. Any file intrinsic except <b>FUPDATE</b> can be called for this file. The EOF is not changed; the record pointer starts at 0. This option is not valid for message files.
0101	Allows update access only if the file's security provisions allows both read and write access. If both read and write access are not allowed, the access type specified in the security provisions (either read or write) is allowed. All file intrinsics, including <b>FUPDATE</b> , can be called for this file. The EOF is not changed; the record pointer starts at 0. This option is not valid for message files.
0110	Allows execute access only if the file's security provisions allow execute access. This access allows read/write access to any loaded file. The program must be running in PM to specify execute access. This option is not valid for message files.
1000	Reserved for the operating system.
1001	Allows directory read access. This access allows you to open a directory and read its contents. Attempt to open a file with this type of access will return an error.

## FOPEN

Bits	Value/Meaning
11:1	<p>Multirecord</p> <p>Indicates whether or not individual read or write requests are confined to record boundaries.</p> <p>The following bit settings are valid:</p> <p>0 Nonmultirecord mode (<b>NOMULTI</b>)</p> <p>1 Multirecord mode (<b>MULTI</b>)</p> <p>Default: 0</p> <p>If the number of half words or bytes to be transferred (specified in the <i>length</i> parameter of the read or write request) exceeds the size of the physical record that is referenced, the remaining half words or bytes are taken from subsequent successive records until the number specified by <i>length</i> has been transferred. For message (MSG) files not accessed with the copy mode option (<i>aoption</i> bit (3:1)) enabled, the file system sets this option to zero. This option is available only if the inhibit buffering option (<i>aoption</i> bit (7:1)) is set to 1.</p>



**Bits      Value/Meaning**

10:1      Dynamic locking

Enables/disables file locking for the file. When this option is specified, the **FLOCK** and **FUNLOCK** intrinsics can be used to dynamically permit or restrict concurrent access to a disk file by other processes at specified times.

The following bit settings are valid:

0      Disallow dynamic locking/unlocking

1      Allow dynamic locking/unlocking

Default: 0

The process can continue this temporary locking/unlocking until it closes the file. If several accessors are sharing the file, they must all specify, or not specify, this option. For example, if a file is opened with the dynamic locking option enabled, and a subsequent accessor tries to open the file with dynamic locking disabled, the subsequent attempt to open fails.

Dynamic locking/unlocking is made possible through the equivalent of a global resource identification number (RIN) assigned to the file and temporarily acquired by **FOPEN**.

Cooperating accessors that have opened a file with the dynamic locking option enabled must access the file using either the **FLOCK** and **FUNLOCK** intrinsics to ensure exclusive use of the file. These accessors are allowed concurrent access even when not using **FLOCK** and **FUNLOCK**, but exclusiveness is not guaranteed.

Lock access is available to a process if it has lock, execute, append, or write access to the file. This option is ignored for files not residing on disk.

If this option is specified for a new file, the dynamic locking bit is NOT changed to 0 as it is on MPE V systems. When a file is new, there can be only one accessor so setting this bit really makes no sense. When opening a directory, dynamic locking must be set to 0 (disallowed).

(ASC) Not valid for asynchronous device files.

## FOPEN

Bits	Value/Meaning
------	---------------

8:2	Exclusive option
-----	------------------

Indicates continuous exclusive access to this file, from open to close. Use this option when performing a critical operation (for example, updating the file).

The following bit settings are valid:

- 00 If access type option (*aoption* bits (12:4)) specifies read only access, then read-share access takes effect. Otherwise, exclusive access takes effect. Regardless of which access option was selected, **FFILEINFO** reports zero.
- 01 Exclusive access. After the file is opened, any additional **HPFOPEN/FOPEN** requests for this file are prohibited until this process issues the **FCLOSE** request or terminates. If any process is already accessing this file when an **HPFOPEN/FOPEN** call is issued with exclusive access specified, an error status is returned. If another **HPFOPEN/FOPEN** call is issued for this file while exclusive access is in effect, an error code is returned to the process that issued the call. Request exclusive access only if the lock access mode is allowed by the security provisions for the file.

For message files, specifying this value indicates that there can be only one writer and only one reader.

Exclusive access cannot be specified for directories.

Bits	Value/Meaning
10	<p>Read-share access (semi-exclusive access). After the file is opened, concurrent write access to this file through another <b>HPFOPEN/FOPEN</b> request is prohibited, whether issued by this process or another process, until this process issues the <b>FCLOSE</b> request or terminates. A subsequent request for the read/write or update access type option (<i>aoption</i> bits (12:4)) obtains read access. However, other types of read access are allowed. If a process already has write access to the file when this call is issued, an error code is returned to the calling process.</p> <p>If another <b>HPFOPEN/FOPEN</b> call that violates the read only restriction is issued while read-share access is in effect, that call fails and an error code is returned to the calling process. Request read-share access only if the lock access mode is allowed by the security provisions for the file.</p> <p>For message files, this value specifies there can be multiple writers, but only one reader.</p>
11	<p>Share access. After the file is opened, concurrent access to this file by any process is permitted, in any access mode, subject to other security provisions in effect.</p> <p>For message files, this value specifies that there can be multiple readers and multiple writers.</p>

Default: 00

## FOPEN

Bits	Value/Meaning
7:1	<p>Inhibit buffering option</p> <p>Enables/disables automatic buffering by the operating system. If <b>NOBUF</b> is specified, I/O is allowed directly between the data area and the applicable hardware device.</p> <p>The following bit settings are valid:</p> <p>0 Allow normal buffering (<b>BUF</b>)</p> <p>1 Inhibit buffering (<b>NOBUF</b>)</p> <p>Default: 0</p> <p><b>NOBUF</b> access is for physical block transfer; not logical record transfer. If <b>NOBUF</b> is specified with this option as well as a variable-length record structure in record format option (<i>foption</i> bits (8:2)) and the file does not have a variable-length record format, then the format is changed internally to an undefined-length record format. Therefore, you are responsible for buffer management. When performing an <b>FWRITE</b>, the variable structure must be set up.</p> <p><b>NOBUF</b> access assumes responsibility for blocking and deblocking of records in the file. To be consistent with files built using buffered I/O, records should begin on half word boundaries. When the information content of the record is less than the defined record length, pad the record with blanks if the file is ASCII, or with zeros if the file is binary.</p> <p>The record size and block size for files with <b>NOBUF</b> specified follow the same rules as those files that are created using buffering. The default blocking factor for a file created under <b>NOBUF</b> is 1.</p> <p>If a file is opened <b>NOBUF</b> without multirecord mode specified in multirecord option (<i>aoption</i> bit (11:1)), a maximum of only one block of data per read or write can be transferred.</p>

Bits	Value/Meaning
	The end-of-file (EOF) marker, next record pointer, and record transfer count are maintained in terms of logical records for all files. The number of logical records affected by each transfer is determined by the size of the transfer.
	Transfers always begin on a block boundary. Those transfers that do not transfer whole blocks leave the next record pointer set to the first record in the next block. The EOF marker always points at the last record in the file.
	For files opened <b>NOBUF</b> , the <b>FREADDIR</b> , <b>FWRITEDIR</b> , and <b>FPOINT</b> intrinsics treat the <i>recnum</i> parameter as a block number.
	Indicate non-RIO access to an RIO file by specifying the file <b>NOBUF</b> . Use the physical block size from <b>FFILEINFO</b> to determine the maximum transfer length.
	For message files, the file system normally resets the inhibit buffering option bit to zero. However, a message file can be opened with <b>NOBUF</b> if the copy mode option ( <i>aoption</i> bit (3:1)) is set to 1; this determines whether access to the file is record-by-record or by block.
	If you are reading a message file with the copy mode option enabled, the inhibit buffering option has the following meaning:
	0    Read by logical record
	1    Read by physical block
	If writing to a message file, open it <b>NOBUF</b> ; if the copy mode option is enabled, access the file block-by-block.
	(ASC) Not valid for asynchronous device files.

## FOPEN

Bits	Value/Meaning
5:2	<p>Multiaccess mode option</p> <p>Indicates how the file's record pointer is to be shared. This option is useful for sharing standard input devices where there is some natural sequence of access to the file. This option permits processes located in different jobs or sessions to open the same file and share that file's record pointer.</p> <p>The following bit settings are valid:</p> <ul style="list-style-type: none"><li>00 No MULTI access. A unique record pointer is created for this access to the file.  For message files, the default value of bits (5:2) is 10 for normal access, or is 00 for copy mode access.</li><li>01 Intrajob MULTI access only allowed. A record pointer is shared with all other opened files of the same name in the same job/session who opened the file with this option set to either 01 or 10.</li><li>10 Interjob MULTI access allowed. A record pointer is shared with all other opened files of the same name on the system.</li></ul> <p>Default: 0</p> <p>(ASC) Not valid for asynchronous device files.</p>
4:1	<p>Nowait I/O option</p> <p>Enables/disables nowait I/O. Determines whether or not the accessor has control returned before the completion of an I/O request. The nowait I/O implies the inhibit buffering option (<i>aoption</i> bit (7:1)); if <b>NOBUF</b> is not specified, the file system does it for you and multirecord access is not available. This option is not available if the file is located on a remote computer. When opening nonmessage files, the process must be running in privileged mode to specify this option.</p> <p>The following bit settings are valid:</p> <ul style="list-style-type: none"><li>0 Nowait I/O not in effect</li><li>1 Nowait I/O in effect</li></ul> <p>Default: 0</p> <p>Nowait I/O cannot be specified for directories.</p>

Bits	Value/Meaning
3:1	<p>Copy mode option</p> <p>Determines whether a file should be treated as a standard sequential file (copy by logical record) or physical block (copy to another file). Copy must be set to obtain EXCLUSIVE access. This causes the multiaccess bits to be set to 00. The following values are valid:</p> <p>0 The file is accessed as its own file type.</p> <p>1 The file is treated as a standard (STD) file, with variable-length records. For message files, this allows nondestructive reading of an old message file at either the logical record or physical-block-record level. Only block-level access is permitted if the file has message file format. This prevents incorrectly formatted data from being written to the message file while it is unprotected.</p> <p>In order to access a message file in copy mode, a process must be able to obtain EXCLUSIVE access to the file. If opening the message file for read only, the file system tries to grant exclusive access; for write only access to the message file, the EXCLUSIVE bits (8:2) in the <i>aoptions</i> must be set to 01.</p> <p>Default: 0</p>
0:3	Reserved for the operating system.

## FOPEN

*resize*

### 16-bit signed integer by value (optional)

Passes the size, in half words or bytes, of the logical records in the file. If *resize=0*, the system uses the default record size, 256 bytes. Positive values are half words; negative values are bytes. The valid range is dependent on storage and record formats:

- For fixed-length and undefined-length ASCII files, the valid range is 1..32,767 bytes.
- For variable-length ASCII files and fixed-length, variable-length, and undefined-length binary files, the range is 1..32,766 bytes (1..16,383 half words). All odd values specified are rounded up to the next even value (the next half word boundary).

Default: Device dependent; refer to the *Accessing Files Programmer's Guide* (32650-90017) for default record sizes.

*device*

### character array (optional)

Passes a string of ASCII characters terminating with any nonalphanumeric excluding a slash (/) or period (.), designating the *device* where the file is to reside. This parameter optionally specifies density for tape files (;DEN= parameter) and/or environment files for certain Hewlett-Packard laser printers (;ENV= parameter), a remote environment (*node#*), and whether the device is to be allocated on a remote machine (;VTERM).

Remote environments (*device=node#*) cannot be specified for directories, byte streams, or when the formal file designator specifies an escaped pathname.

Syntax of the options available through the *device* parameter are:

$$device = [ node\# ] \left[ \begin{array}{l} * \\ **vol \\ *volclass \\ devname \\ devclass \end{array} \right]$$

[ ;DEN=*density* ]

[ ;ENV=*printenv* ]

[ ;VTERM ]



*node#* remote environment option

Passes a remote node name to be accessed through remote file access (RFA). The node name string must be followed by a pound character (#) to delimit the string. The internal format of the name must match that defined by the NS 3000/XL subsystem. Refer to the *NS 3000/XL User/Programmer Reference Manual (36920-90001)*.

Default: Local file access

\* volume set option

Passing only an asterisk character (\*) indicates to FOPEN that the disk file can span all volumes in the volume set bound to the group where the file resides. No set name is actually passed. This option is applicable only at file creation.

\*\**vol* volume name option

Passing a name, preceded by two asterisks (\*\*), indicates that the disk file is restricted to the volume whose name is specified in *vol*. The specified volume must reside within the volume set bound to the group where the file resides. This option is applicable only at file creation.

\**vol-*  
*class* volume class option

Passing a name, preceded by one asterisk (\*), indicates that the disk file is restricted to the volume class whose name is specified in *volclass*. The specified volume class must reside within the volume set bound to the group where the file resides. This option is applicable only at file creation.

## FOPEN

*dev-* device name option

*name*

Passes the logical device number, in ASCII format, of a specific device. The file is assumed to be permanent. If the logical device number of a nonshareable device is specified, the nonshareable device must be ready prior to the **FOPEN** call.

If the device name option is used to open a file residing on a disk, specify the logical device number of the disk drive where a volume which resides within the volume set bound to the group where the file resides is mounted (applicable only at file creation). Thereafter, the disk file is associated with that volume, not with the originally specified logical device number.

*dev-* device class option

*class*

Passes a device class name or volume class name of up to eight alphanumeric characters beginning with an alpha character. For example, **TAPE** or **DISC**. If *devclass* is specified, the file is allocated to any available device in that class.

If the device class option is used to open a file residing on a disk, specify a device class name that corresponds to a valid and mounted volume class name; otherwise, it is assumed that the attempt is to open a device file not a disk file. Applicable only at file creation. The specified volume class must reside within the volume set bound to the group where the file resides. The file is free to span any of the volumes that fall within the specified volume class.

---

### Note

If either *volumeset*, *vol*, *volclass*, *devname*, or *devclass* is not specified, **FOPEN** defaults to a disk file located on the volume class **DISC** bound to the group where the file resides. The file is free to span all volumes in the volume class **DISC**.

When opening a magnetic tape that is shared for a second time, open the device with the device name option instead of device class option. This ensures that the system operator is not confused by a second tape request.

---

**;DEN=** tape density option

*density* Passes the tape density to be used while writing to a tape file. This option is applicable only when writing to a tape on a drive that supports more than one density. When reading from a tape, the density of the tape overrides the density value specified.

To specify density when writing to the tape file, use the keyword **;DEN=**.

Default: The highest density available on the device opened

**;ENV=** printer environment option

*print-  
env* Passes the name of a file that contains a printer environment. Valid only for specified printers. If opening an Hewlett-Packard 268x page printer file, an optional printing environment can be specified for the job. The printing environment is defined as all of the characteristics of the printed page that are not part of the data itself, including the page size, the margin width, the character set, the orientation (horizontal or vertical), and the name of the forms to use. If an environment file is not specified, the file system selects a default printer environment.

To specify a printer environment, assign the keyword **;ENV=**, followed by the name of the environment file, to the *device* parameter in the form **;ENV=*printenv***. Any environment selected remains active until replaced by a new environment or a call **FCLOSE** to close the printer.

Default: No printer environment file specified

## FOPEN

**;VTERM** reverse VT option

Indicates that the specified device name is allocated on a remote machine. Specify the remote location of a device with the *device* parameter as **VTERM**. Specify the remote environment in the same open request, using either the *formaldesig* parameter or the remote environment option (see *node#* above) of the *device* parameter. Reverse VT is almost the same as a terminal opened up through remote file access, except that no session is required on the remote machine.

Default: No reverse VT

*formmsg*

**character array (optional)**

Passes a forms message that can be used for sending messages (for example, telling the system operator what type of paper to use in the line printer) The message is a string of ASCII characters terminated by a period. The maximum number of characters allowed is 49, including the terminating period; more than 49 characters are truncated.

Used for tape label information if bit (6:1) of the *foption* parameter is set. The tape label information is formatted as follows:

```
. [valid[,type[,exp[,seq]]]];
```

The period is required so that the tape label information is not mistaken for a forms message.

*valid* Volume identifier, consisting of <=6 printable characters. In a multivolume set, specify only the first *valid*.

*type* Label type, identified by three alphabetic characters:

ANS - ANSI standard labels (default)  
IBM - IBM standard labels

*exp* Month/day/year of the file expiration date or the date when the file information is no longer valid. The file can be overwritten after this date. If the default is 00/00/00, the file can be overwritten immediately. In a volume set, file expiration dates must be equal to or earlier than the date on the previous file.

- seq* Use one of the following methods to specify the position of the file in relation to other files on the tape:
- A zero, which causes a search of all volumes until the file is found.
  - An unsigned integer (1-9999), which specifies the position of the file relative to the current file on the tape.
  - **ADDF** which causes the tape to be positioned to add a new file on the end of the volume (or last volume in a multivolume set).
  - **NEXT** which positions the tape at the next file on the tape. If this is not the first **FOPEN** for the file and a rewind occurred on the last close, then the position remains at the beginning of the previous file.

Default: **NEXT**

(**KSAM**) Contains a description of the primary key and up to 15 alternate keys:

- If a new file is being created, this parameter must be specified.
- If this is an existing file, check flagword field to see if the default values are acceptable; if not acceptable, specify this field explicitly.

For **KSAM XL** files, refer to the *Using KSAM XL* (32650-90168) for a complete description of this parameter.

For **KSAM/3000** files, refer to the *KSAM/3000 Reference Manual* (30000-90079) for a complete description of this parameter.

## FOPEN

- userlabels* **16-bit signed integer by value (optional)**  
Passes the number, in the range 0..254, of user-label records to be created for the file. Applicable to new disk files only.  
Default: 0  
(ASC) Not valid for asynchronous device files.
- blockfactor* **16-bit signed integer by value (optional)**  
Passes the number of logical records in one physical record (block). This value is used to calculate the physical record size for disk and magnetic tape files. The valid range for this option is 1..255. If a value greater than 255 is specified, FOPEN resets the value to 255. If a value less than 1 is specified, FOPEN resets the value to the default *blockfactor* setting.  
For fixed-length records, *blockfactor* specifies the actual number of records in a block. For variable-length records, *blockfactor* is interpreted as a multiplier used to compute the block size (*resize\*blockfactor*). For undefined-length records, *blockfactor* is always one logical record per block.  
Default: For files opened **BUF**, the default is calculated by dividing the specified *resize* into the physical record size (block size) configured for the device. For files opened **NOBUF**, the default is 1.  
(ASC) Not valid for asynchronous device files.

*num-  
buffer*    **16-bit signed integer by value (optional)**

Passes the number of buffers, number of copies, and output priority indicated by setting the following bits:

<b>Bits</b>	<b>Value/Meaning</b>
-------------	----------------------

11:5	Numbuffers
------	------------

Indicates the number of buffers to allocate to the file. The valid range is dependent on file type:

- For standard files, the valid range is 1..31.
- For circular and RIO files, the valid range is 1..16.
- For message files, the valid range is 2..16 (if a 1 is specified, the file system sets this option to 2 and no error is returned).
- For KSAM/3000 files, the valid range is 1..20.
- Not used for KSAM XL files.

This option is ignored for standard disk files, useful only for slow devices (such as tapes) used in a buffered mode, and is not applicable for files representing interactive terminals.

This option must not specify a number of buffers whose combined size exceeds the physical capacity of the file.

Default: 00010 (decimal 2)

4:7	Spooler copies
-----	----------------

Indicates the number of copies of the file to be produced by the spooling facility. The valid range is 0 ... 127. This option is applicable to spooled devices only. Specify this option for a file already opened (for example, **\$STDLIST**), this is the highest value supplied before the last **FCLOSE** takes effect. The copies do not appear continuously if the system operator intervenes or if a file of higher output priority is ready before the last copy is complete.

Default: 0000001 (decimal 1)

## FOPEN

Bits	Value/Meaning
------	---------------

0:4	Output priority
-----	-----------------

Indicate the output priority to be attached to the file for spooled output. Applicable only to spooled devices. The valid range for this option is the binary equivalent of 1..13. The output priority must be a number between 1 (lowest priority) and 13 (highest priority), inclusive. If a value less than the current outfence set by the system operator is specified, file printing is deferred until the operator raises the output priority of the file or lowers the outfence. Specify this option for a file already opened (for example, `$STDLIST`), the highest value supplied before the last `FCLOSE` takes effect.

Default: 1000 (decimal 8)

*filesize*

### 32-bit signed integer by value (optional)

Passes the maximum file capacity. For variable-length records, the capacity is expressed in blocks (block = *recordsize* \* *blockfactor*). For fixed-length and undefined-length records, the capacity is expressed in logical records:

- The maximum file size for standard and KSAM XL files is 4,294,901,759 bytes (4 gigabytes less 64K bytes).
- The maximum file size for RIO, circular, and message files is dependent upon both the record size and number of extents defined for the file:
  - For circular and RIO files, a maximum file size of 500 megabytes is possible if *recsize*=256 bytes and *numextent*=32.
  - For message files, a maximum file size of 500 megabytes is possible if *recsize*=128 bytes and *numextent*=32.

Applicable only at file creation.

Default: 1023 records

(ASC) Not valid for asynchronous device files.



- numextent*      **16-bit signed integer by value (optional)**
- Passes a value in the range 1..32 that determines the number of extents for the file. If a value of 1 and an *initialloc* value of 1 is specified, the file is created as one contiguous extent of disk space. If a value >1 is specified, a variable number of extents (with varying extent sizes) is allocated on a need basis. Applicable only at file creation.
- Default: >8 extents
- (ASC) Not valid for asynchronous device files.
- initialloc*      **16-bit signed integer by value (optional)**
- Passes an integer value in the range 1..32 that determines the number of extents to be allocated to the file initially. Applicable only at file creation.
- Default: 1
- (ASC) Not valid for asynchronous device files.
- filecode*      **16-bit signed integer by value (optional)**
- Passes a value that can be used as a file code to identify the type of file. This code is recorded in the file label and is accessible through the **FFILEINFO** intrinsic. Applicable only at file creation (except when opening an old file that has a negative file code).
- If the program is running in user mode, specify a file code in the range 0..32,767 to indicate the file type being created; programs running in user mode can access files with non-negative file codes. If the program is running in privileged mode, specify a file code in the range -32,768..32,767; programs running in privileged mode can access files with a file code in the range -32,768..32,767. If an old file with a negative file code is opened, the file code specified must match the file code in the file label.
- Negative file codes are only allowed for file in MPE groups.
- Default: 0
- (ASC) Not valid for asynchronous device files.

**FOPEN**

**Table 4-9. FOPEN/HPFOPEN Parameter Equivalents**

FOPEN Parameter	HPFOPEN Itemnum,Item
<i>flenum</i> (functional return)	<i>flenum</i> (parameter)
<i>formaldesig</i>	2, <i>formaldesig</i>
<i>foption</i> : Bits (14:2) Domain Bit (13:1) ASCII/binary Bits (10:3) File designator Bits (8:2) Record format Bit (7:1) Carriage-control Bit (6:1) Labeled tape Bit (5:1) Disallow file equation Bits (2:3) File type	3, <i>domain</i> 53, <i>ASCII/binary</i> 5, <i>file designator</i> 6, <i>record format</i> 7, <i>carriage-control</i> 8, <i>labeled tape</i> 9, <i>disallow file equation</i> 10, <i>file type</i>
<i>aoption</i> : Bits (12:4) Access type Bit (11:1) Multirecord Bit (10:1) Dynamic locking Bits (8:2) Exclusive Bit (7:1) Inhibit buffering Bits (5:2) Multiaccess mode Bit (4:1) Nowait I/O Bit (3:1) File copy	11, <i>access type</i> 15, <i>multirecord</i> 12, <i>dynamic locking</i> 13, <i>exclusive</i> 46, <i>inhibit buffering</i> 14, <i>multiaccess mode</i> 16, <i>nowait I/O</i> 17, <i>file copy</i>
<i>recsize</i>	19, <i>record size</i>
<i>device</i>	20, <i>device name</i> 22, <i>volume class</i> 23, <i>volume name</i> 24, <i>density</i> 25, <i>printer environment</i> 26, <i>remote environment</i> 42, <i>device class</i> 48, <i>reverse VT</i>
<i>formmsg</i>	8, <i>labeled tape label</i> 28, <i>spooled message</i> 30, <i>labeled tape type</i> 31, <i>labeled tape expiration</i> 32, <i>labeled tape sequence</i> 54, <i>KSAM parms</i>
<i>userlabels</i>	33, <i>user labels</i>
<i>blockfactor</i>	40, <i>block factor</i>

Table 4-9. FOPEN/HPFOPEN Parameter Equivalents (continued)

FOPEN Parameter	HPFOPEN Itemnum,Item
<i>numbuffers:</i>	
Bits (11:5) Numbuffers	<i>44, numbuffers</i>
Bits (4:7) Spooler copies	<i>34, spooler copies</i>
Bits (0:4) Output priority	<i>27, output priority</i>
<i>filesize</i>	<i>35, filesize</i>
<i>numextent</i>	<i>47, numextent</i>
<i>initialloc</i>	<i>36, initial allocation</i>
<i>filecode</i>	<i>37, filecode</i>

### Operation Notes

A file can be referenced by its formal file designator. When executed, a unique file number is returned to the process. This file number can be used, rather than the formal file designator, in other calls to this file.

---

**Note** If a file is opened with read access only (bits 12:4 = 0000) a subsequent call to FOPEN to write and set eof to 0 (bits 12:4 = 0001) succeeds but only after the system changes the call to write but not set eof to 0 (bits 12:4 = 0010). This protects the reader, who opened the file first, from having the eof set to 0.

---

### Condition Codes

CCE (2) Request granted. The file is open.

CCG (0) Not returned.

CCL (1) Request denied. For example, another process already has exclusive or semi-exclusive access for this file, the privilege level of this file is not user (3), or an initial allocation of disk space cannot be made due to lack of disk space. If the file is not opened successfully, the file number value returned by FOPEN is 0. Call the FCHECK intrinsic for more details.

### Related Information

Intrinsics HPFOPEN

Commands None

Manuals *Accessing Files Programmer's Guide* (32650-90017), *Using KSAM XL* (32650-90168), and *KSAM/3000 Reference Manual* (30000-90079)

---

## FPARSE

NM and CM callable.

Parses and validates file designators. Only MPE syntax names can be parsed. Escaped or POSIX syntax names will return errors.

### Syntax

CA      I16A   U16A   I32A FPARSE( <i>formaldesig</i> , <i>result</i> , <i>item</i> , <i>vector</i> );
---

### Parameters

*formaldesig*    **character array (required)**

Passes the file reference string to be parsed. Terminate the string with any nonalphanumeric character except a slash (/), period (.), or colon (:).

If an error occurs, the first element of the *vector* array returns the byte offset of the invalid item in *formaldesig*.

*result*        **16-bit signed integer array (required)**

Returns two elements; the first element is the error number, the second element is reserved and always returns 0.

A positive value indicates the syntax of the specified file reference is correct and what type of file reference is being made. A negative value indicates a syntax error in the file reference.

The possible error number values are:

Value	Meaning
0	Regular file designator
1	Backreference (* is the first character in <i>formaldesig</i> )
2	System file (\$) is the first character in <i>formaldesig</i> )

The default designator numbers for system files, as defined for FOPEN *foption*, are:

0	File name
1	\$STDLIST
2	\$NEWPASS
3	\$OLDPASS
4	\$STDIN
5	\$STDINX
6	\$NULL

Value	Meaning
-1	Invalid item values
-2	Parameter bounds violation
-3	Illegal delimiter; misuse of ., /, or :
-4	<i>Item</i> specified without specifying <i>vector</i>
-5	Illegal value in <i>item</i>
-6	Item list not terminated by 0
-7	Undefined system file
-8	Lockword not allowed in backreferenced (*) file
-9	NS 3000/XL not present; <i>envidname</i> was specified
-101	First character of the file name not alphabetic
-102	File name expected in the string
-103	File name identifier too long
-104	First character of lockword not alphabetic
-105	Lockword expected in string
-106	Lockword identifier too long
-107	First character of group name not alphabetic
-108	Group name expected in string
-109	Group name identifier too long
-110	First character of the account name not alphabetic
-111	Account name expected in string
-112	Account name identifier too long
-113	First character of <i>envidname</i> not alphabetic
-114	Expected <i>envidname</i> in string
-115	Identifier for <i>envidname</i> too long

---

**Note** Up to three identifiers can be specified, separated by a period (.), in an *envidname*; errors -113 through -115 can reference one of the three identifiers. The error pointer indicates the location.

If NS 3000/XL is not installed on the system, FPARSE returns a -9 for file designators with an *envidname* after the file name. This error indicates that everything up to *envidname* was valid, but a parsed *vector* array is not returned.

---

*items* **16-bit unsigned integer array (optional)**

Passes an item code, one value per element, indicating what the corresponding *vector* parameter points to:

## FPARSE

Value	Meaning
0	End of item list
1	File name
2	Lockword
3	Group name
4	Account name
5	NS 3000/XL <i>envidname</i>

*vector*      **32-bit signed integer array (optional)**  
Returns vectors, one per element, for the requested *item*.

---

**Note**      A *vector* cannot be passed without a corresponding *item* value.

---

Information is returned as follows:

Bits	Value/Meaning
16:16	Return the length of the string specified by the corresponding <i>items</i> parameter.  If zero is returned, the requested item in <i>formaldesig</i> was not specified.
0:16	Return the offset, in bytes, from the base of the <i>formaldesig</i> parameter to the first element of the string segment specified by the corresponding <i>item</i> parameter.

The last element returns the file designator length passed in *formaldesig* (bits (16:16)) and the default type (bits (0:16)) if the *result* parameter indicates a system file.

---

**Note**      It is not recommended that applications parse file names on their own. Instead, use the FPARSE intrinsic to do so.

---

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017) and the <i>NS 3000/XL User/Programmer Reference Manual</i> (36920-90001)

---

## 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).

### Syntax

I16V    I32V FPOINT( <i>filenum</i> , <i>lrecnum</i> );
--

### Parameters

*filenum*    **16-bit signed integer by value (required)**

Passes the file number of the file where the pointer is to be set.

*lrecnum*    **32-bit signed integer by value (required)**

Passes the relative logical record (or block number for a **NOBUF** file) where the logical record pointer is to be positioned. The number of the first logical record is 0.

On disk files, the EOF indicator is the file limit.

If the *filenum* parameter refers to a byte stream file, the *lrecnum* parameter is interpreted as a byte offset to which the file pointer is set.

(KSAM) For fixed-length files, record numbering starts with 0 or 1, depending on how the file was created.

(KSAM/3000) For KSAM/3000 files with variable-length records, the number of the first logical record is 0.

## FPOINT

### Condition Codes

- CCE (2) Request granted.
- CCG (0) Request denied. The logical record pointer position is unchanged. Positioning was requested at a point beyond the file limit.
- CCL (1) Request denied. The logical record pointer position is unchanged because of one of the following:
- *Lrecnum* <0
  - Invalid *filenum* parameter
  - I/O is pending on a nowait request
  - Spooled file
  - Not a direct access disk file
  - File does not contain fixed-length or undefined-length records
  - Not allowed with append access
- (KSAM) The *filenum* parameter was invalid or *lrecnum* specified a record marked for deletion.

### Operation Notes

Not applicable to message files.

### Related Information

- Intrinsics None
- Commands None
- Manual *Accessing Files Programmer's Guide* (32650-90017), *Using KSAM XL* (32650-90168), and *KSAM/3000 Reference Manual* (30000-90079)



---

## FREAD

NM and CM callable.

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

### Syntax

I16	I16V	UDS	I16V
<i>transfercount</i> :=FREAD( <i>filenum</i> , <i>buffer</i> , <i>length</i> );			

### Functional Return

*transfercount*    **16-bit signed integer (assigned functional return)**

Returns the length of the data transferred to *buffer*:

- If a negative value is passed in the *length* parameter, *transfercount* is a positive value indicating the number of bytes transferred.
- If a positive value is passed in the *length* parameter, *transfercount* is a positive value indicating the number of half words transferred.
- If nowait I/O is specified in FOPEN/HPFOPEN, a zero is returned. The record length is returned in the *length* parameter of the IOWAIT or IODONTWAIT intrinsic.

---

**Note**            If the file was opened with the nowait I/O option enabled, *transfercount* is zero. The IOWAIT intrinsic can be used to determine the length of the data transferred.

---

### Parameters

*filenum*            **16-bit signed integer by value (required)**

Passes the file number of the file to be read. If *filenum* references a directory node, FREAD fails with a CCL file code.

*buffer*             **user-defined structure (required)**

Returns the record that was read. This structure must be large enough to hold all of the information to be transferred.

## FREAD

*length*      16-bit signed integer by value (required)

Passes the length of the data to be transferred to *buffer*:

Value	Meaning
<0	Length in bytes
=0	No transfer occurs
>0	Length in half words

If *length* is larger than the size of the logical record, and the multirecord *aoption* in **FOPEN/HPFOPEN** was not specified, transfer is limited to the length of the logical record. If the multirecord *aoption* in **FOPEN/HPFOPEN** is specified, transfer continues until either *length* is satisfied or end-of-data is encountered, and each transfer begins at the start of the next physical record (block). Any data remaining in the last physical record read is inaccessible. Data transfer for byte stream files continues until either *length* bytes are read or an end-of-file occurs.

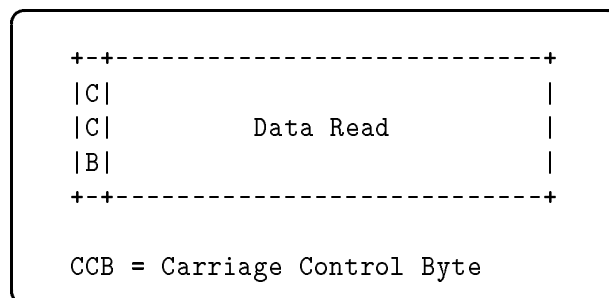
### Operation Notes

When the logical end-of-data is encountered, **CCG (0)** is returned to the process:

- On magnetic tape, the end-of-data is a physical indicator such as a tape mark. When the program reads a file that spans more than one volume of labeled magnetic tape, the program is suspended until an operator mounts the next tape. **CCG (0)** is not returned when end-of-tape is encountered.
- On disk, the end-of-data occurs when there is an attempt to read past the last logical record of the file. In this case, **CCG (0)** is returned, and no record is read.
- If the file is embedded in an input source containing commands, an end-of-data occurs when an **EOD** command is encountered. End-of-data is indicated by a hardware end-of-file, including **EOF** (on **\$STDIN**), any record beginning with a colon (:), or on **\$STDINX**, by **EOD**.
- On a standard input device for a job (not a session) **JOB**, **E0J**, or **DATA** indicate end-of-data.

When reading from an empty message file and another process has opened that file for write access, the process waits. If the message file is empty and there are no writers, the process waits if there is an `FCONTROL=45` in effect or if this is the first `FREAD` after the reader's `FOPEN/HPFOPEN`. Otherwise, `CCG (0)` is returned. If an `FREAD` is issued against a message file and an `FCONTROL=46` is in effect, the writer's ID and the record type code are appended to the beginning of the record.

When an old file containing carriage-control characters supplied through the *controlcode* parameter of the `FWRITE` intrinsic is read, and you specified either the carriage-control *foption* parameter of the `FOPEN/HPFOPEN` intrinsic, or the `CCTL` parameter of the `FILE` command, the carriage-control byte is read as follows:



**Figure 4-8. Carriage Control Byte**

### Condition Codes

- `CCE (2)` Request granted. The information was read.
- `CCG (0)` Request denied. The logical end-of-data was encountered during reading. When reading a labeled magnetic tape file that spans more than one volume, `CCG (0)` is not returned when end-of-tape (EOT) is encountered. `CCG (0)` is returned at actual end-of-file, with a transmission log of 0 if there is an attempt to read past end-of-file.
- `CCL (1)` Request denied. The information was not read because a terminal read was terminated by a special character or timeout interval, as specified in the `FCONTROL` intrinsic, or a tape error was recovered and the `FSETMODE` option was enabled. Check the condition codes both in normal I/O and in `nowait` I/O.

### Related Information

- Intrinsics      None
- Commands        None
- Manuals        *Accessing Files Programmer's Guide* (32650-90017)

---

## 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.

### Syntax

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

### Functional Return

*transfercount*    **16-bit signed integer**

Returns the length of the data transferred to *buffer*:

- If a negative value is passed in the *length* parameter, the *transfercount* is a positive value indicating the number of bytes transferred.
- If a positive value is passed in the *length* parameter, the *transfercount* is a positive value indicating the number of half words transferred.

---

### Note

If the file is opened with the nowait I/O option enabled, *transfercount* is zero. The IOWAIT intrinsic can be used to determine the length of the data transferred.

---

### Parameters

*filenum*        **16-bit signed integer by value (required)**

Passes the file number of the file to be read.

*buffer*        **user-defined structure (required)**

Returns the specified record. This structure must be large enough to hold all of the information to be returned.

***length*** 16-bit signed integer by value (required)

Returns the length of the data to be transferred to *buffer*:

Value	Meaning
<0	Length in bytes
=0	No transfer occurs
>0	Length in half words

If *length* is larger than the size of the logical record, and the multirecord *aoption* in FOPEN/HPFOPEN was not specified, transfer is limited to the length of the logical record. If the multirecord *aoption* in FOPEN/HPFOPEN is specified, transfer continues until either *length* is satisfied or end-of-data is encountered, and each transfer begins at the start of the next physical record (block). Any data remaining in the last physical record read is inaccessible.

### Operation Notes

Three restrictions apply when using FREADBACKWARD:

- Used only with a magnetic tape drive with read-reverse capability.
- The magnetic tape drive must be located on an HP-IB system.
- The magnetic tape NOBUF must be accessed.

---

**Note** Magnetic tape devices with read-reverse capability are not supported. A call to FREADBACKWARD results in an error (CCL (1)).

---

Not used for KSAM files.

### Condition Codes

CCE (2)	Request granted.
CCG (0)	Request denied. The logical beginning-of-data was encountered. When reading a labeled magnetic tape file that spans more than one volume, CCG (0) is not returned when beginning-of-tape (BOT) is encountered. CCG (0) is returned at actual beginning of file, with a transmission log of 0 if an attempt is made to read past beginning-of-file.
CCL (1)	Request denied. The information was not read because a tape error occurred, or a tape error was recovered and the FSETMODE option was enabled. Check the condition code in both normal I/O and nowait I/O.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

---

## FREADBYKEY

NM and CM callable.

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

---

**Note** For KSAM files only.

---

### Syntax

```
I16V          I16V  LA  I16V  CA
length:=FREADBYKEY(file num,target,tcount,keyvalue,);
                I16V
                keylocation);
```

### Functional Return

*length*      **16-bit signed integer by value (assigned functional return)**

Returns the length of the information transferred. If *tcount* is positive, *length* is a word count; if negative it is a byte count.

### Parameters

*file num*      **16-bit signed integer by value (required)**

Passes the file number of the file to be read.

*target*      **logical array (required)**

Contains the transferred record. It must be large enough to hold all the information to be read.

*tcount*      **16-bit signed integer by value (required)**

Passes the number of words or bytes to be transferred. If *tcount* is positive, it is the length in words; if negative, it is the length in bytes.

If *tcount* is less than the size of the record to be transferred, only the first *tcount* words or bytes are transferred from the record. If the *tcount* is larger than the physical record size, only the physical record length is transferred.

***keyvalue***      **character array (required)**

Passes the value determining the record to be read. The first record found with an identical value in the key specified by ***keylocation*** is the record read.

***keylocation***    **16-bit signed integer by value (required)**

Passes the relative byte location in the record of the key whose value determines which record is to be read. The first byte is numbered as 1. If a value of zero is specified, the primary key is used.

### Condition Codes

CCE (2)      Request granted.

CCG (0)      Request denied. The logical end-of-data or beginning-of-data was encountered during the read.

CCL (1)      Request denied. An error occurred. Either an I/O error occurred or the key could not be located.

### Related Information

Intrinsics      None

Commands      None

Manuals      *Using KSAM XL (32650-90168) and KSAM/3000 Reference Manual (30000-90079)*

---

## FREADC

NM and CM callable.

FREADC returns records in physical sequence, not chronological sequence. For KSAM/XL files with the REUSE option, physical is not equal to chronological sequence and FREADC reads the file in physical sequence. Deleted records are not returned using the FREADC intrinsic, the FREADDIR intrinsic must be used if deleted records are required.

---

**Note** For KSAM files only.

---

### Syntax

I16V	I16V	LA	I16V
<i>length</i> := FREADC( <i>filenum</i> , <i>target</i> , <i>tcount</i> );			

### Functional Return

*length*      **16-bit signed integer by value (assigned functional return)**  
Returns the length of the information transferred. If *tcount* is positive, it is a word count; if negative, it is a byte count.

### Parameters

*filenum*      **16-bit signed integer by value (required)**  
Passes the file number of the file to be read in chronological sequence.

*target*      **logical array (required)**  
Returns the transferred record. It must be large enough to hold all the information to be read.

*tcount*      **16-bit signed integer by value (required)**  
Passes the number of words or bytes to be transferred. If *tcount* is positive, it is the length in words; if negative, it is the length in bytes.  
  
If *tcount* is less than the size of the record to be transferred, only the first *tcount* words are transferred from the record. If the *tcount* is larger than the physical record size, only the physical record length is transferred.



**Condition Codes**

CCE (2)	Request granted.
CCG (0)	Request denied. The logical end-of-data or beginning-of-data was encountered during the read.
CCL (1)	Request denied. An error occurred.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Using KSAM XL (32650-90168) and KSAM/3000 Reference Manual (30000-90079)</i>

---

## 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.

### Syntax

I16V    UDS    I16V    I32V FREADDIR( <i>filenum</i> , <i>buffer</i> , <i>length</i> , <i>lrecnum</i> ) ;
--

### Parameters

- filenum*    **16-bit signed integer by value (required)**  
Passes the file number of the file to be read.
- buffer*    **user-defined structure (required)**  
Returns the record that was read. This structure should be large enough to hold all of the information to be transferred.
- length*    **16-bit signed integer by value (required)**  
Passes the number of half words or bytes to be transferred. If this value is positive, it signifies half words; if negative, bytes. Zero signifies that no transfer occurs. If *length* is less than the size of the logical record, only the first *length* half words or bytes are read from the record.  
If *length* is larger than the size of the logical record and the multirecord *aoption* in FOPEN/HPFOPEN was not specified, the transfer is limited to the length of the logical record. If the multirecord *aoption* in FOPEN/HPFOPEN is specified, the remaining half words or bytes specified in *length* are read from the next records.
- lrecnum*    **32-bit signed integer by value (required)**  
The relative number, in the file, of the logical record or block number for the NOBUF files to be read. A value of zero indicates the first logical record.  
(KSAM) Indicates the relative chronological record number where the the chronological pointer is positioned. Chronological record numbering for fixed-length records starts with 0 or 1, as specified when the file was built.

## Operation Notes

This intrinsic is different than the **FREAD** intrinsic. The **FREAD** intrinsic reads only the record already pointed to by the logical record pointer. Issue the **FREADDIR** intrinsic only for disk files composed of fixed-length or undefined-length records. If RIO access is used, **FREADDIR** inputs the specified logical record. If the record is inactive, the contents of the inactive record are transmitted and CCE (2) is returned; there is no indication of the block containing some inactive records. (**FCHECK** returns a nonzero error number to distinguish active and inactive records.) If an RIO file is accessed using the non-RIO method (**NOBUF**), **FREADDIR** inputs the specified block.

After the **FREADDIR** intrinsic is executed, the logical record pointer is set to the beginning of the next logical record, or the first logical record of the next block for **NOBUF** files.

It is possible to skip portions of records inadvertently, if the multirecord *option* of **FOPEN/HPFOPEN** is set and the *length* parameter specified is greater than one logical record. For example, if all of record 11 and half of record 12 is read in a file, the logical record pointer is set to the beginning of record 13 after the **FREADDIR** intrinsic executes; the second half of record 12 is skipped.

Byte stream files are applicable with **FREADDIR** intrinsic. The *lrecnum* parameter specifies a byte offset in the byte stream file, which is equivalent to a logical record number, since the record size is one byte.

Not applicable to message files.

## Condition Codes

CCE (2)	Request granted. The information was read.
CCG (0)	Request denied. End-of-file was encountered.
CCL (1)	Request denied. The information was not read; an error occurred.

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017), <i>Using KSAM XL</i> (32650-90168), and <i>KSAM/3000 Reference Manual</i> (30000-90079)

---

## FREADLABEL

NM and CM callable.

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

### Syntax

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

### Parameters

- filenum*     **16-bit signed integer by value (required)**  
Passes the file number of the file whose label is to be read.
- buffer*     **user-defined structure (required)**  
Returns the label that was read. This structure must be large enough to hold the number of half words specified by *length*.
- length*     **16-bit signed integer by value (optional)**  
Passes the number of half words to be transferred from the label, <=128.  
Default: 128 half words
- labelid*    **16-bit signed integer by value (optional)**  
Passes the label number. For labeled tapes, *labelid* is ignored. The next sequential label is read.  
Default: Zero

### Operation Notes

When a disk file is opened, user labels can be read from, or written to, in any order, at any time, regardless of access capabilities to the rest of the file. A disk file can have <=254 128-half word user-defined labels.

A magnetic tape file (if labeled) must be labeled with an ANSI-standard or IBM-standard label. The operating system automatically skips over any unread user-defined labels when the first **FREAD** intrinsic call for files is issued. To read a user-defined label, you should call the **FREADLABEL** intrinsic immediately after an **FOPEN/HPFOPEN** intrinsic has opened the file. The user-defined label must be 40 half words in length to conform to the length of the ANSI-standard or IBM-standard label.

**Condition Codes**

CCE (2)	Request granted. The label was read.
CCG (0)	Request denied. A label was referenced beyond the label written on the file.
CCL (1)	Request denied. The label was not read; an error occurred.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

---

## FREADSEEK

NM and CM callable.

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

### Syntax

I16V    I32V FREADSEEK( <i>file num</i> , <i>lrec num</i> );
---

### Parameters

*file num*    **16-bit signed integer by value (required)**

Passes the file number of the file to be read.

*lrec num*    **32-bit signed integer by value (required)**

Passes the relative number of the logical record to be read. A value of zero indicates the first logical record.

### Operation Notes

The FREADSEEK intrinsic enhances direct access of disk files. When a certain record is required, use FREADSEEK before a call to FREADDIR to transfer it to the stack. The FREADSEEK intrinsic directs the operating system to move the record from disk into a buffer in anticipation of the FREADDIR call, which moves the record directly to the stack.

Issue the FREADSEEK intrinsic call only for direct-access files where I/O buffering and fixed-length or undefined-length records are in effect.

FREADSEEK fails with a CCL condition code when *file num* references a NOBUF file. Byte stream files are opened NOBUF, and not applicable to FREADSEEK intrinsic.

Not applicable to message files.

(KSAM) This intrinsic cannot be used for KSAM files. If called for a file created as a KSAM file, CCL (1) is returned.

### Condition Codes

CCE (2)    Request granted.

CCG (0)    Request denied. A logical end-of-file indication was encountered.

CCL (1)    Request denied. An error occurred.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

---

## FREEDSEG

NM and CM callable.

Releases an extra data segment assigned it by the `GETDSEG` intrinsic. Data segment management (DS) capability is required.

---

**Note** Data segment management (DS) intrinsics are not recommended for use in the native mode programming environment. Use of DS intrinsics in NM will degrade your program's performance.

---

### Syntax

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

### Parameters

- index*      **16-bit unsigned integer by value (required)**  
Passes the index assigned to the data segment, obtained from a `GETDSEG` intrinsic call.
- id*          **16-bit unsigned integer by value (required)**  
Passes the identification assigned to the segment. Enter zero if none is assigned.

### Operation Notes

If this is a private data segment or a shareable (nonprivate) segment not currently assigned to any other process in the job/session, the segment is deleted from the entire job/session. If it is a shareable segment that is currently assigned to another process, it is deleted from the calling process but continues to exist in the job/session.

If the process is running in privileged mode when the extra data segment is acquired, then the process must be in privileged mode when it invokes `FREEDSEG`.



**Condition Codes**

- CCE (2) Request granted. The data segment is deleted from the job/session.
- CCG (0) Request granted. The data segment is deleted from the calling process but continues to exist in the job/session; it is being shared by another process.
- CCL (1) Request denied. Either the *index* is invalid, or *index* and *id* do not specify the same shared data segment.

**Related Information**

- Intrinsics None
- Commands None
- Manuals *Introduction to MPE XL for MPE V Programmers (30367-90005)*

---

## **FREELOCRIN**

NM and CM callable.

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

### **Syntax**

```
FREELOCRIN;
```

### **Operation Notes**

The **FREELOCRIN** intrinsic must be called before a second call to **GETLOCRIN**.

### **Condition Codes**

- |         |  |
|---------|--|
| CCE (2) | Request granted.   |
| CCG (0) | Request denied. No RINs are currently reserved for the job.            |
| CCL (1) | Request denied. A minimum of one RIN is currently locked by a process. |

### **Related Information**

- |            |   |
|------------|---|
| Intrinsics | None  |
| Commands   | None  |
| Manuals    | <i>Resource Management Programmer's Guide (32650-90024)</i> |

## FRELATE

NM and CM callable.

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

### Syntax

U16	I16V	I16V
<i>intordup</i> := FRELATE( <i>infilenum</i> , <i>listfilenum</i> );		

### Functional Return

*intordup*      **16-bit unsigned integer (assigned functional return)**

Returns values indicating whether the two files referenced are interactive and/or duplicative:

Bits	Value/Meaning
15:1	0    Files specified by <i>infilenum</i> and <i>listfilenum</i> do not form an interactive pair.
	1    Files specified by <i>infilenum</i> and <i>listfilenum</i> form an interactive pair.
0:1	0    Files specified by <i>infilenum</i> and <i>listfilenum</i> do not form a duplicative pair.
	1    Files specified by <i>infilenum</i> and <i>listfilenum</i> form a duplicative pair.

### Parameters

*infilenum*      **16-bit signed integer by value (required)**

Passes the file number of the input file.

*listfilenum*    **16-bit signed integer by value (required)**

Passes the file number of the list file.

## FRELATE

### Operation Notes

A device file is interactive if it requires human intervention for all input operations; this is necessary to establish the person/machine dialog required to support a session.

A device file is duplicative if all input operations are echoed to a corresponding display without intervention by the operating system software.

The interactive/duplicative attributes of a file pair do not change between the time the files are opened and closed.

(KSAM) This intrinsic cannot be used for KSAM files. If called for a file created as a KSAM file, CCL (1) is returned.

### Condition Codes

- CCE (2) Request granted.
- CCG (0) Request denied. *Infilenum* and/or *listfilenum* corresponds to \$NULL (a logical file that contains no data). No data can be read from this file, and all data written to it is discarded. Interactive or duplicative functions do not apply.
- CCL (1) Request denied. An error occurred.

### Related Information

- Intrinsics None
- Commands None
- Manuals *Accessing Files Programmer's Guide* (32650-90017) and *Getting System Information Programmer's Guide* (32650-90018)

---

## FREMOVE

NM and CM callable.

Marks the current record in a KSAM file for deletion.

---

**Note** For KSAM files only.

---

### Syntax

I16V  
FREMOVE(*file num*)

### Parameters

*file num* 16-bit signed integer by value (required)

Passes the file number of the file where the record is to be deleted.

### Operation Notes

Split stack calls are permitted.

Deleted records can be read with this intrinsic.

(KSAM/3000) When executed, the first word of the current record is set to 1's; the record is not physically deleted from the file. When the file is read by a KSAM read intrinsic, the deleted records are not read.

(KSAM/XL) When executed, the first bit in the record header is set to 1.

### Condition Codes

CCE (2) Request granted.

CCG (0) Request denied. The logical end-of-data was encountered.

CCL (1) Request denied. An error was encountered or the record does not contain the requested key value; the record is not deleted.

### Related Information

Intrinsics None

Commands None

Manuals *Using KSAM XL* (32650-90168) and *KSAM/3000 Reference Manual* (30000-90079)

---

## 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.

### Syntax

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

### Parameters

*filenum*      **16-bit signed integer by value (required)**

Passes the file number of the file to be renamed.

*formaldesig*    **character array (required)**

Passes a formal file designator interpreted according to MPE-escaped semantics. The file name must be terminated by a nonalphanumeric character other than a period (*.*), a slash (*/*), a hyphen (*-*), and an underscore (*\_*).

The file referred to by *formaldesig* can be either an MPE file (i.e., one that uses MPE syntax) or it can follow HFS syntax. If *formaldesig* follows MPE syntax, the file name can include password, group, and account in the following format:

*filename/lockword.group.account*

To keep or add a lockword to the file, enter it in the ASCII string. If you do not specify a group, the file will remain in the same group it was assigned before you invoked **FRENAME**. If you are renaming a new or temporary file, specify any account that shares the same volume set as the file referred to by *formaldesig*.

If *formaldesig* follows HFS syntax, the file name must start with either a dot (*.*) or a slash (*/*).

If *formaldesig* is an escaped pathname:

- you cannot reference remote files
- it cannot express a name equivalent to *filename:envid*
- you cannot use the *device* parameter (*device=node#*) to specify the remote location of a device

If *formaldesig* is the name of a user-defined file, it can begin with an asterisk (\*). If *formaldesig* is the name of a system-defined file, it can begin with a dollar sign (\$). When creating a KSAM file, *formaldesig* must be a unique file name, that is, one not currently existing in the permanent file directory.

The formal file designator can contain command interpreter variables and expressions that are evaluated before *formaldesig* is parsed and validated.

The file referred to by *formaldesig* may reside either in an MPE group or in an HFS directory.

If renaming a file, a process must have the following:

TD	Traverse directory entry to access to all directories specified in <i>formaldesig</i> . If <i>formaldesig</i> is specified as <code>file.group.account</code> , the directories are the root directory, the account, and the MPE group. If there is no TD access, <b>FRENAME</b> fails and a call to <b>FCHECK</b> with <i>flenum</i> set to 0 will return a system error code (398) in the <i>fserrorcode</i> parameter.
CD	Create directory entry to access to the new parent directory.
DD	Delete directory entry to access to the old parent directory.
SF	Save files capability.

The file referred to by *formaldesig* may reside either in an MPE group or in an HFS directory.

If a file is referenced by *flenum*, you can rename it within the hierarchical directory as long as the process invoking **FRENAME** has sufficient access and the restrictions are satisfied. **FRENAME** intrinsic fully qualifies the file owner name. Only file owners and users with appropriate privilege can manipulate a file's lockword.

The home volume set of *formaldesig* must be the same as the file being renamed. Volume sets cannot be spanned when renaming files.

## Operation Notes

The following restrictions apply to **FRENAME**:

- Directories cannot be renamed.
- Lockwords cannot be assigned to hierarchical directories.
- Files cannot be renamed across volume sets.
- Files with KSAM/3000, RIO, and CIR file types may only be assigned names in the MPE name space.

If a file without an ACD is renamed from an MPE group to a directory (although not within the same account), an ACD is automatically assigned to the file.

All errors will set the condition codes to CCL.

(KSAM) Although **FRENAME** will work on CM KSAM files, it renders them unuseable until they are reassigned their original names. To rename a CM KSAM file, use the **KSAMUTIL RENAME** command.

## **FRENAME**

### **Condition Codes**

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred.

### **Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)



---

## 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.

### Syntax

<pre style="margin: 0;">          I16V    U16V FSETMODE(<i>filename</i>,<i>modeflags</i>) ;</pre>
---

### Parameters

*filename*      **16-bit signed integer by value (required)**

Passes the file number of the file whose access modes are to be set.

*modeflags*    **16-bit unsigned integer by value (required)**

Passes the access mode option:

Bits	Value/Meaning
15:1	Controls chronological order of user program writes. This mode controls write requests to the file from all writers on the system.
0	Do not change the write queue where this file is being accessed
1	Access this file through the serial write queue (ASC) Not used for asynchronous serial devices. (KSAM) Set this bit to zero.

---

**Note**      A file placed on the serial write queue remains until the file is purged.

---

## FSETMODE

Bits	Value/Meaning
14:1	<p>Controls blocking of program execution until completion of physical write operations. This mode controls writes to <i>filenum</i> only from the calling process.</p> <p>0 Return control to the program from a write request without waiting for completion of the physical write operation</p> <p>1 Force the program to wait until the physical write operation is completed (the record is posted)</p> <p>(ASC) Not used for asynchronous serial devices.</p> <p>(KSAM) Bit (14:1) only is returned. The possible values are:</p> <p>1 Activate output verification: All output to the file is verified as physically complete before an <b>FWRITE</b>, <b>FUPDATE</b>, or <b>FREMOVE</b> intrinsic returns control to the user. When a logical record is written, a CCE (2) condition is returned.</p> <p>0 Deactivate output verification: Output is not verified.</p>
13:1	<p>Controls issuing line feed to a terminal after each terminal read.</p> <p>0 Issue a line feed at the completion of each read from a terminal.</p> <p>1 Inhibit issuance of a line feed at the completion of each read from a terminal.</p> <p>(KSAM) Set this bit to zero.</p>

Bits	Value/Meaning
12:1	Reporting tape device automatic error recovery.
0	Do not report automatic error recovery on a tape device.
1	Report tape device automatic error recovery by returning CCL (1) to <b>FREAD</b> and <b>FWRITE</b> .
	(ASC) Not used for asynchronous serial devices.
	(KSAM) Set this bit to zero.
0:12	Reserved for the operating system

## Operation Notes

There are two ways to service a program's write request to a file:

- In an order that maximizes overall system throughput, using the normal write queue.
- In the order that the program requested the writes, using the serial write queue. Write requests placed on the serial write queue are guaranteed to be completed in chronological order.

FSETMODE intrinsic fails with a CCL condition:

- If *filenum* references a byte stream file.
- If FCHECK intrinsic is called to obtain the error code associated with a CCL condition code.

## Condition Codes

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred.

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

---

## 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.

### Syntax

I16V      I16V FSPACE( <i>filenum</i> , <i>displacement</i> );
---

### Parameters

*filenum*      **16-bit signed integer by value (required)**

Passes the file number of the file on which spacing is to be done.

*displacement*      **16-bit signed integer by value (required)**

Passes the number of logical records for buffered disk files, or blocks for NOBUF files and all tape files, to be spaced over, relative to the current position of the logical record pointer.

A positive value signifies forward spacing, a negative value signifies backward spacing. The maximum positive value is 32,767, the maximum negative value is -32,768.

If RIO access is used, *displacement* includes both active and inactive records. Attempts to backspace beyond the beginning of the file are ignored by the system. The logical record pointer points to record 0 (the first record), and no error codes are returned.

### Operation Notes

The FSPACE intrinsic cannot be used with variable-length record files, message files, or with spooled files on disk. An attempt to use this intrinsic on such files results in CCL (1), and the logical record pointer is left at its current position.

FSPACE intrinsic fails with a CCL condition:

- If *filenum* references a byte stream file.
- If FCHECK intrinsic is called to obtain the error code associated with a CCL condition code.

(KSAM) The logical record pointer is repositioned in key sequence. The spacing is based on the primary key unless an alternate key has been specified in a prior call to FFINDN, FFINDBYKEY, or FREADBYKEY.

**Condition Codes**

- CCE (2) Request granted.
- CCG (0) Request denied. An end-of-file indicator was encountered during spacing. For disk files, this is the file limit, and the logical record pointer is not changed. For magnetic tape files, this is the end-of-file mark, and the logical record pointer points to the (logical) end-of-file. The magnetic tape is positioned to one record past the file mark on the tape. For labeled tape, the logical record pointer is at the file mark.
- CCL (1) Request denied. An error occurred; for example, the file resides on a device that prohibits spacing. Not allowed with append access.

**Related Information**

- Intrinsics None
- Commands None
- Manuals *Accessing Files Programmer's Guide* (32650-90017), *Using KSAM XL* (32650-90168), and *KSAM/3000 Reference Manual* (30000-90079)

---

## FUNLOCK

NM and CM callable.

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

### Syntax

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

### Parameters

*filenum*      **16-bit signed integer by value (required)**  
Passes the file number of the file whose global RIN is to be unlocked.

### Condition Codes

CCE (2)      Request granted.  
CCG (0)      Request denied. The calling process did not lock the file.  
CCL (1)      Request denied. The file was not opened with the dynamic locking *option* of the FOPEN/HPFOPEN intrinsic, or the *filenum* parameter is invalid.

### Related Information

Intrinsics      None  
Commands      None  
Manuals      *Resource Management Programmer's Guide* (32650-90024)

## FUPDATE

NM and CM callable.

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

### Syntax

```

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

### Parameters

*filenum*     **16-bit signed integer by value (required)**

Passes the file number of the file to be updated.

*buffer*     **user-defined structure (required)**

Passes the record to be written in the update.

*length*     **16-bit signed integer by value (required)**

Passes the number of half words or bytes to be written to the file. A positive value is in half words; a negative value is in bytes.

For files opened **BUF**:

- If *length* ≤ record size, the length is transferred in half words or bytes and remaining portions of the record are padded with fill characters.
- If *length* = 0, no transfer occurs, and the record address is overwritten with default fill characters (blanks for ASCII files and null characters for binary files).
- If *length* > record size, CCL (1) is returned and no transfer occurs.

For files opened **NOBUF**:

- If *length* ≤ block size, the length is transferred in half words or bytes and remaining portions of the record are padded with fill characters.
- If *length* > block size, CCL (1) is returned and no transfer occurs.

### Condition Codes

CCE (2)     Request granted.

CCG (0)     Request denied. An end-of-file condition was encountered during updating.

CCL (1)     Request denied. An error occurred; the file is not residing on disk, or *length* exceeds the size of the block when multirecord mode is not in effect.

## FUPDATE

### Operation Notes

This intrinsic affects the logical record (or block for **NOBUF** files) of the file last referenced by any intrinsic call, except an **FPOINT** call, which affects the record prior to the last record referenced. **FUPDATE** moves the specified information from the stack into this record. The file containing this record must be opened with the update *option* specified in the **FOPEN/HPFOPEN** call and the file cannot have variable-length records. If **RIO** access is used, the modified record is set to the **ACTIVE** state.

**FUPDATE** intrinsic fails with a **CCL** condition:

- If *filename* references a byte stream file.
- If **FCHECK** intrinsic is called to obtain the error code associated with a **CCL** condition code.

The **FUPDATE** intrinsic is not applicable to message files.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)



## 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.

### Syntax

I16V    UDS    I16V    U16V FWRITE( <i>filenum</i> , <i>buffer</i> , <i>length</i> , <i>controlcode</i> );
---

### Parameters

- filenum*    **16-bit signed integer by value (required)**  
 Passes the file number of the file to be written on.
- buffer*    **user-defined structure (required)**  
 Passes the record to be written.
- length*    **16-bit signed integer by value (required)**  
 Passes the number of half words or bytes to be written to the record. If this value is positive, it signifies half words; if negative, bytes. Zero indicates that no transfer occurs.
- If *length* is less than the *recsize* parameter associated with the record, only the first *length* (half words or bytes) is written.
- If *length* is larger than the logical record size and **NOBUF** (*aoption* in **FOPEN/HPFOPEN**) is not specified, the **FWRITE** request is refused and **CCL (1)** is returned. If **NOBUF** is specified, *length* cannot exceed the physical record size unless the multirecord *aoption* is specified also.
- If the multirecord *aoption* in **FOPEN/HPFOPEN** is specified, the excess half words or bytes are written to succeeding physical records and carriage-control is specified, the actual data transferred is limited to *recsize* minus one byte.
- (ASC) For asynchronous devices, the physical data transfer is byte-by-byte. If half words are specified, an even number of bytes is output.

## **FWRITE**

*controlcode* **16-bit unsigned integer by value (required)**

Passes a carriage-control code, effective if the file is transferred to a line printer or terminal (including a spooled file whose destination is a line printer or a terminal). This parameter is effective only for files opened with carriage-control specified.

The options are:

<b>Value</b>	<b>Meaning</b>
0	Print the full record transferred, using single spacing. This results in a maximum of 132 characters per printed line.
1	Use the first character of the data written to satisfy space control and suppress this character on the printed output. This results in a maximum of 132 characters of data per printed line.

Use any octal code from Table 4-10 to determine space control and print the full record transferred. This results in a maximum of 132 characters per printed line.

If the *controlcode* parameter is not 0 or 1, and *length* is 0, only the space control is executed and no data is transferred.

Determine whether the carriage-control directive takes effect before printing (prespace movement) or after printing (postspace movement), with the **FCONTROL** intrinsic.

For spooled files, use **FWRITE** instead of **FCONTROL** to set the prespace/postspace control and the auto/no auto page eject control; use control codes %100 through %103 and %400 through %403 for this. If one of the above controls is specified with *length*=0, no physical I/O occurs.

When a file is opened with CCTL, use the carriage-control codes in one of the following ways:

- As the value of the *controlcode* parameter.
- When *controlcode*=1, as the first byte of the *buffer* array.

The default carriage-control code is postspacing with automatic page eject. This applies to all Hewlett-Packard subsystems except FORTRAN 77/XL and COBOLII/XL (these have prespacing with automatic page eject).

---

**Note**

Channel assignments shown in Table 4-10 are the Hewlett-Packard standard defaults.

---

(KSAM) This parameter must be specified to satisfy internal requirements, but it is ignored.

**Table 4-10. Carriage-Control Directives**

Octal Code (ASCII)	Description of Carriage Action
%2 - %52 (" ")	Single space (with or without automatic page eject)
%53 ("+")	No space (next printing at column 1), cannot be used more than once on the Hewlett-Packard 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")	<p>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:</p> <p>Postspace mode The current request has a transfer count of 0 and the previous request was <b>FOPEN</b>, <b>HPFOPEN</b>, <b>FCLOSE</b>, or <b>FWRITE</b> specifying a carriage-control directive of %61.</p> <p>Prespace mode The previous request has a transfer count of 0, and the current request and previous request are any combination of <b>FOPEN</b>, <b>HPFOPEN</b>, <b>FCLOSE</b>, or <b>FWRITE</b> specifying a carriage-control directive of %61.</p>
%62	Skip to one line before top of form (valid for Hewlett-Packard 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 Hewlett-Packard 2608S and 2563A printers only)
%62 - %77 (" ")	Single space (with or without automatic page eject; for terminals and serial printers)
%104 - %177 (" ")	Single space (with or without automatic page eject; for terminals and serial printers)
%2nn	Space nn lines (no automatic page eject); nn is any octal number from 0 through 77
%300 - %313	Select VFC Channel 1 - 12 (Hewlett-Packard 2613, 2617, 2618, 2619)
%300 - %317	Select VFC Channel 1 - 16 (Hewlett-Packard 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
%305	Skip to next 1/2 page
%306	Skip to next 1/4 page
%307	Skip to next 1/6 page
<b>4-258 Command Definitions</b>	Skip to bottom of form

**Table 4-10. Carriage-Control Directives (continued)**

Octal Code (ASCII)	Description of Carriage Action
%311	User option (Hewlett-Packard 2613/17/18/19), skip to one line before bottom of form (Hewlett-Packard 2608A/S)
%312	User option (Hewlett-Packard 2613/17/18/19), skip to one line before top of form (Hewlett-Packard 2608A/S)
%313	User option (Hewlett-Packard 2613/17/18/19), skip to top of form (Hewlett-Packard 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	(Hewlett-Packard 2607)
%314 - %317	(Hewlett-Packard 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	Set 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	Set prespace movement option (spaces first, then prints)
%402 or %102	Set single-space option, with automatic page eject (60 lines per page)
%403 or %103	Set 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 Hewlett-Packard 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.

FWRITE

FOPEN or :FILE	FWRITE Controlcode												
	= 0	= 1	= Greater than 1										
Carriage Control Foption or CCTL	<table border="1"> <tr> <td>Byte 1</td> <td>resize 133</td> </tr> <tr> <td>0</td> <td>record = 132</td> </tr> </table> <p>Data output contains 132 characters; the prefix byte is added and contains 0.</p>	Byte 1	resize 133	0	record = 132	<table border="1"> <tr> <td>resize 132</td> </tr> <tr> <td>record = 132</td> </tr> </table> <p>Data output contains 132 characters; the control character in the first byte is not printed if output is to a list device.</p>	resize 132	record = 132	<table border="1"> <tr> <td>Byte 1</td> <td>resize 133</td> </tr> <tr> <td>control</td> <td>record = 132</td> </tr> </table> <p>Data output contains 132 characters; the prefix character added is a carriage control character specified by the FWRITE control parameter</p>	Byte 1	resize 133	control	record = 132
Byte 1	resize 133												
0	record = 132												
resize 132													
record = 132													
Byte 1	resize 133												
control	record = 132												
Carriage Control Foption not specified or NOCTL	<table border="1"> <tr> <td>132</td> </tr> <tr> <td>record = 132</td> </tr> </table> <p>Data output contains 132 character.</p>	132	record = 132	<table border="1"> <tr> <td>132</td> </tr> <tr> <td>record = 132</td> </tr> </table> <p>Data output contains 132 characters.</p>	132	record = 132	<table border="1"> <tr> <td>132</td> </tr> <tr> <td>record = 132</td> </tr> </table> <p>Data output contains 132 characters.</p>	132	record = 132				
132													
record = 132													
132													
record = 132													
132													
record = 132													

Effect on Data Output

LG200154\_004

Figure 4-9. Carriage-Control Data

## Operation Notes

If information is written to a fixed-length record and the `NOBUF` *option* in `FOPEN/HPFOPEN` is not specified, any unused portion of the record is padded with binary zeros or ASCII blanks.

When the `FWRITE` intrinsic is executed, the logical record pointer is set to the record immediately following the record just written, or to the first logical record in the next block for `NOBUF` files. If `RIO` access is used, the modified record is set to the `ACTIVE` state.

When an `FWRITE` call writes a record beyond the current logical end-of-file indicator, this indicator is advanced. However, this is noted in the file label only when the file is closed or when an extent is allocated. If the physical bounds of the file are reached, `CCG (0)` is returned.

If a magnetic tape is unlabeled (as specified in the `FOPEN/HPFOPEN` intrinsic, or in the `FILE` command) and the program attempts to write over or beyond the physical or simulated end-of-tape (EOT) marker, the `FWRITE` intrinsic returns `CCL (1)`. The actual data is written to the tape, and a call to `FHECK` reveals an end-of-tape file error. After the EOT marker is crossed, all writes to the tape transfer the data successfully but return `CCL (1)` until the tape crosses the EOT marker again in the reverse direction (rewind or backspace).

If a magnetic tape is labeled (as specified in the `FOPEN/HPFOPEN` intrinsic, or in the `FILE` command), `CCL (1)` is not returned when the tape passes the EOT marker. Attempts to write to the tape after an EOT marker cause end-of-volume (EOV) markers to be written. A message is printed on the system console requesting another volume (reel of tape) be mounted.

For message files and circular files, this intrinsic logically appends the record to the end of the file. If a circular file is full, the first block is deleted, the remaining blocks are logically shifted to the file's head, and the new record is appended to the end of the file. If a message file is full and there are no readers, the process waits if there is an `FCONTROL=45` in effect or if this is the first `FWRITE` after an `FOPEN/HPFOPEN` call. Otherwise, `CCG (0)` is returned.

Control codes are ignored for byte stream files. Byte stream files can not be opened with carriage control.

## **FWRITE**

### **Condition Codes**

- CCE (2) Request granted.
- CCG (0) Request denied. The physical bounds of the file prevented further writing.
- CCL (1) Request denied. An error occurred; *length* exceeded the size of the record in nonmultirecord mode, the **FSETMODE** option was enabled to signify recovered tape errors, or the end-of-tape marker was sensed. If the file is being written to a multivolume labeled magnetic tape set, CCL (1) is not returned when the end-of-tape marker is sensed. Instead, end-of-volume labels are written, and a request is issued to mount the next volume.

### **Related Information**

- Intrinsics None
- Commands None
- Manuals *Accessing Files Programmer's Guide* (32650-90017)



## FWRTEDIR

NM and CM callable.

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

### Syntax

```

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

### Parameters

- filenum*     **16-bit signed integer by value (required)**  
 Passes the file number of the file to be written to.
- buffer*     **user-defined structure (required)**  
 Passes the record to be written. This structure should be large enough to hold all of the information to be transferred.
- length*     **16-bit signed integer by value (required)**  
 Passes the number of half words or bytes to be written to the file. A positive value is in half words; a negative value is in bytes.
- For files opened **BUF**:
- If *length* ≤ record size, the length is transferred in half words or bytes and remaining portions of the record is padded with fill characters.
  - If *length* = 0, no transfer occurs and the record address is overwritten with default fill characters (blanks for ASCII files; null characters for binary files).
  - If *length* > record size, CCL (1) is returned and no transfer occurs.
- For files opened **NOBUF** and **nonmultirecord**:
- *Length* is transferred in half words or bytes up to the file limit (ignores all block boundaries) and remaining portions of the last block written to is padded with fill characters.
- lrecnum*     **32-bit signed integer by value (required)**  
 Passes the relative number of the logical record, or block number for **NOBUF** files, to be written. Zero indicates the first record.

## **FWRITEDIR**

### **Operation Notes**

This intrinsic differs from the **FWRITE** intrinsic, the **FWRITE** intrinsic writes only the record pointed to by the logical record pointer. Use the **FWRITEDIR** intrinsic only for disk files composed of fixed-length or undefined-length records.

When information is written to a fixed-length record and **NOBUF** is not specified in the **FOPEN/HPFOPEN** call that opens the file, any unused portion of the record is padded with binary zeros or ASCII blanks.

When the **FWRITEDIR** intrinsic is executed, the logical record pointer is set to the record immediately following the record just written, or the first logical record of the next block for **NOBUF** files.

If **RIO** access is used, the modified record is set to the **ACTIVE** state.

When an **FWRITEDIR** call writes a record beyond the current logical end-of-file indicator, the indicator is advanced. This can result in the creation of dummy records to pad the records between the previous end-of-file and the newly written record. When the new record is in the same extent, these dummy records are filled with binary zeros or with ASCII blanks.

When the physical bounds of the file prevent further writing because all allowable extents are filled, the end-of-file condition (**CCG (0)**) is returned to the program.

Not applicable to message files.

(**KSAM**) This intrinsic cannot be used for **KSAM** files. If called for a file created as a **KSAM** file, **CCL (1)** is returned.

### **Condition Codes**

- |                |   |
|----------------|---|
| <b>CCE (2)</b> | Request granted.  |
| <b>CCG (0)</b> | Request denied. The physical end-of-file was encountered. |
| <b>CCL (1)</b> | Request denied. An error occurred.                        |

### **Related Information**

- |            |   |
|------------|---|
| Intrinsics | None  |
| Commands   | None  |
| Manuals    | <i>Accessing Files Programmer's Guide (32650-90017)</i> |

## FWRITE LABEL

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.

### Syntax

```

I16V  UDS  I16V  I16V
FWRITE LABEL(filename,buffer,length,labelid);
```

### Parameters

- filename*     **16-bit signed integer by value (required)**  
 Passes the file number of the file to be labeled.
- buffer*        **user-defined structure (required)**  
 Passes the label to be written. If the file is a labeled magnetic tape file, this label must be 40 half words in length.
- length*        **16-bit signed integer by value (optional)**  
 Passes the number of half words or bytes passed by *buffer*. A positive value is in half words; a negative value is in bytes.
- labelid*       **16-bit signed integer by value (optional)**  
 Passes the number of the label to be written. The first label is zero. This parameter is ignored for labeled tapes. The next sequential tape label is written. The default is zero.

### Operation Notes

Once a disk file is opened, it is possible to read from or write to user-defined labels regardless of the access to the rest of the file. If the file is on labeled magnetic tape, the user-defined label must be 40 half words in length to conform to the length of the ANSI-standard or IBM-standard label.

### Condition Codes

- CCE (2)        Request granted.
- CCG (0)        Request denied. The calling process attempted to write a label beyond the limit specified in the FOPEN/HPFOPEN intrinsic when the file was created.
- CCL (1)        Request denied. An error occurred.

## **FWRITELABEL**

### **Related Information**

Intrinsics      None

Commands      None

Manuals      *Accessing Files Programmer's Guide* (32650-90017), *Using KSAM XL* (32650-90168), and *KSAM/3000 Reference Manual* (30000-90079)

## GENMESSAGE

NM and CM callable (differences noted below).

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

### Syntax

```

I16          I16V   I16V   I16V
msglength := GENMESSAGE(file num, set num, msg num,
                        CA I16V   I16V
                        buffer, buffersize, parmask,
                        *      *      *      *
                        param1, param2, param3, param4
                        *      I16V
                        param5, msg destination,
                        I16
                        error num);

```

### Functional Return

*msglength*     **16-bit signed integer (assigned functional return)**  
Returns the length of the message (in bytes).

### Parameters

*file num*     **16-bit signed integer by value (required)**  
Passes the file number that specifies the message catalog.

*set num*     **Positive 16-bit signed integer by value (required)**  
Passes the message set number within the catalog (>0,<63).

*msg num*     **Positive 16-bit signed integer by value (required)**  
Passes the message number within the message set (>0,<32767).

*buffer*     **character array (optional)**  
Returns the assembled message.

*buffersize*     **16-bit signed integer by value (optional)**  
Returns the length of *buffer*. If *buffer* is specified, the length is in bytes. If *buffer* is not specified, returns the length (in bytes) of the records to be written to the destination file.  
Default: 72 bytes  
(For a single-line message, the combined length of the message number, a blank space separator, and message text must not exceed the default buffer size of 72 bytes.)

## GENMESSAGE

*paramask*      **16-bit signed integer by value (optional)**

Passes a mask indicating parameter types for *param1* through *param5*. The bit settings are:

Bits	Value/Meaning
13:3	<i>param5</i> type: 000 String, terminated by an ASCII null (0) 001 16-bit signed integer 010 32-bit signed integer by reference 011 Ignored
10:3	<i>param4</i> type: 000 String, terminated by an ASCII null (0) 001 16-bit signed integer 010 32-bit signed integer by reference 011 Ignored
7:3	<i>param3</i> type: 000 String, terminated by an ASCII null (0) 001 16-bit signed integer 010 32-bit signed integer by reference 011 Ignored
4:3	<i>param2</i> type: 000 String, terminated by an ASCII null (0) 001 16-bit signed integer 010 32-bit signed integer by reference 011 Ignored
1:3	<i>param1</i> type: 000 String, terminated by an ASCII null (0) 001 16-bit signed integer 010 32-bit signed integer by reference 011 Ignored
0:1	value: 1 Ignore rest of word and parameters <i>param1</i> through <i>param5</i> 0 Rest of word, in 3-bit groupings, specifies parameter types for <i>param1</i> through <i>param5</i>

Default: *param1* through *param5* are ignored.

*param1*  
through  
*param5*

**type varies (optional)**

(NM) Passes the parameter to be inserted into the message. If *paramask* is 000 (string), pass the 32-bit address of the array containing the string. If *paramask* is 001 (16-bit signed integer), pass a signed 16-bit integer (a value from -32,768 through 32,767). If *paramask* is 010 (32-bit signed integer by reference), pass the 16-bit address of the word identifier that returns the value.

(CM) Passes the parameter to be inserted into the message. If *paramask* is 000 (string), pass the 16-bit address of the array containing the string. If *paramask* is 001 (16-bit signed integer), pass a signed 16-bit integer (a value from -32,768 through 32,767). If *paramask* is 010 (16-bit signed integer by reference), pass the 16-bit address of the word identifier that returns the value.

*msgdestination* **16-bit signed integer by value (optional)**

Returns the file number of the message's destination file (0 or -1=\$STDLIST, >2=file number of the destination file).

Default: \$STDLIST if you do not specify *buffer*, no file if you do specify *buffer*.

## GENMESSAGE

*errornum*     **16-bit signed integer by reference (optional)**

Returns one of the following values, indicating the success or failure of the intrinsic call.

Value	Meaning
0	Successful
1	<b>FREADLABEL</b> call on catalog file failed
2	<b>FREAD</b> call on catalog file failed
3	Specified <i>setnum</i> not found in catalog
4	Specified <i>msgnum</i> not found in catalog
6	Message overflowed buffer (if <i>msgdestination</i> specified, message routed correctly)
7	Write to destination file failed
8	Catalog file opened with improper access options
11	<i>flenum</i> was not specified
12	<i>setnum</i> was not specified
13	<i>msgnum</i> was not specified
14	<i>setnum</i> < 0 was specified
15	<i>setnum</i> > 62 was specified
16	<i>msgnum</i> <= 0 was specified
17	<i>bufferize</i> <= 0 was specified
18	<i>msgdestination</i> <= -2 was specified
19	Reference parameter out of bounds

### Operation Notes

The calling program opens a message catalog, allowing the insertion parameters supplied by **GENMESSAGE**. Depending on how the *msgdestination* and *buffer* parameters are defined, **GENMESSAGE** then routes the message to its destination. To route the message to a file, specify *msgdestination*. To return the message to a calling program, specify *buffer*. To do both, specify both *msgdestination* and *buffer*.

Open the catalog file as a permanent ASCII file with the **NOBUF** and **MULTI** access options enabled.



**Condition Codes**

CCE (2)	Request granted.
CCL (1)	Request denied. File system error.
CCG (0)	Request denied. Missing required parameter, invalid parameter, or invalid file number of catalog or destination file.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Message Catalogs Programmer's Guide</i> (32650-90021)

---

## GETDSEG

NM and CM callable.

Creates or acquires an extra data segment for use by the process. Data segment management (DS) capability is required.

---

**Note** Data segment management (DS) intrinsics are not recommended for use in native mode programming environment. Use of DS intrinsics in NM degrades your program's performance.

---

### Syntax

U16    I16    U16V GETDSEG( <i>index,length,id</i> );
--

### Parameters

- index*      **16-bit unsigned integer by reference (required)**
- Returns the index of the extra data segment assigned by the operating system. When GETDSEG is called in user mode, *index* is a logical index of the assigned data segment; if an error is found, *index* is set to %2000-%2004. When GETDSEG is called in privileged mode, *index* is the actual data segment table (DST) entry number assigned for the data segment.
- length*      **16-bit signed integer by reference (required)**
- Passes the user-specified maximum size, in half words, of the newly created data segment. If the data segment already exists, *length* returns the maximum size, in half words, of the data segment.
- id*            **16-bit unsigned integer by value (required)**
- Passes the identity that declares the extra data segment shareable between other processes in the job/session, or private to the calling process. For a shareable extra data segment, specify *id* as a nonzero value. If an extra data segment with the same identification already exists, it is made available to the calling process; otherwise, a new extra data segment, shareable within the job/session, is created with *id*. For a private extra data segment, specify an *id* of zero.

## Operation Notes

The number of extra data segments that can be requested, and the maximum size allowed these segments, is limited by parameters specified when the system is configured. When an extra data segment is created, the **GETDSEG** intrinsic returns a logical index number to the process. This index number is assigned by the operating system and allows the process to reference the extra data segment in later intrinsic calls.

Assign the extra data segment to an identity that either allows other processes in the job or session to share the extra data segment, or that declares it private to the calling process. If the extra data segment is shareable, other processes can obtain its index (through **GETDSEG**) and use this index to reference the extra data segment. Thus, the index is a local name that identifies the extra data segment throughout any process that obtained the index with the **GETDSEG** call. The index does not need to be the same value in all processes sharing the extra data segment. However, the identity is a job-wide or session-wide name that permits any process to determine the index of the extra data segment. If **GETDSEG** is called in user mode, the extra data segment is initially filled with zeros.

When **GETDSEG** is called in user mode, all subsequent calls to intrinsics that use *index* must be in user mode. When **GETDSEG** is called in privileged mode, all subsequent calls to intrinsics that use *index* must be in privileged mode.

When a data segment is created through **GETDSEG**, sufficient virtual space is allocated by the system to accommodate the original length of the data segment. This virtual space is allocated in increments of 512 half words. For example, creation of an extra data segment with a length of 600 half words results in the allocation of 1024 half words of virtual space.

## GETDSEG

### Condition Codes

- CCE (2) Request granted. A new extra data segment was created.
- CCG (0) Request granted. An extra data segment with this identity exists already.
- CCL (1) Request denied. The following values are returned in *index*:

Value	Meaning
%2000	An illegal <i>length</i> was specified.
%2001	The process requested more than the maximum allowable number of data segments.
%2002	Sufficient storage was not available for the data segment.
%2003	A stack expansion necessary to satisfy the request could not be done because the stack was frozen or the stack is already at its maximum size (stack expansion is usually not necessary to get an extra data segment).
%2004	There is not enough room in the job definition table to make an entry for the extra data segment.

### Related Information

- Intrinsics None
- Commands None
- Manuals *Introduction to MPE XL for MPE V Programmers (30367-90005)*

## GETINFO

NM and CM callable.

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

### Syntax

I16	CA	I16	I16
<i>result</i> :=GETINFO( <i>infostring</i> , <i>infolength</i> , <i>parm</i> );			

### Functional Return

*result*      **16-bit signed integer (assigned functional return)**

Returns an execution value:

Value	Meaning
0	Successful execution.
1	Execution error, one of the following conditions exist: <ul style="list-style-type: none"> <li>■ <i>Infostring</i> was specified, but <i>infolength</i> was not specified.</li> <li>■ <i>Infolength</i> was specified, but <i>infostring</i> was not specified.</li> <li>■ Either an invalid <i>infostring</i> or <i>infolength</i> value was specified.</li> <li>■ An invalid <i>parm</i> value was specified.</li> </ul>

### Parameters

*infostring*      *character array (optional)*

Returns an information string (the contents of an array) passed to the calling process by the *info* parameter of the RUN command, or by the item associated with *itemnum* 11 of the CREATEPROCESS intrinsic.

If the information string was not supplied at process creation time, or if it is of length zero, *infostring* is returned unmodified.

If *infostring* is specified, *infolength* must pass the length, in bytes, of *infostring*.

## GETINFO

*infolength*      *16-bit signed integer by reference (optional)*

Passes and receives a value.

If *infostring* is specified, *infolength* must pass the length, in bytes, of *infostring*.

When returning from the call, *infolength* contains the smaller of either the length of the information string returned in *infostring* or the original value of *infolength*.

---

**Note**              Specify a value at least as large as the length of the information string supplied at process creation time. If *infolength* is too small, the information string is truncated upon return to *infostring*.

*Infostring* and *infolength* must be specified as a pair.

---

*parm*              *16-bit signed integer by reference (optional)*

Returns a value passed to the calling process at process creation time by one of the following sources:

- The *parm* parameter of the RUN command.
- The item associated with *itemnum* 2 of the CREATEPROCESS intrinsic.
- The *param* parameter of the CREATE intrinsic.

A zero is returned if none of the above are used to pass a value.

## Operation Notes

Returns user-supplied information to the calling process from one of the following sources:

- If the process was created with the RUN command, the contents of the *info* and *parm* parameters can be retrieved.
- If the process was created with the CREATEPROCESS intrinsic, retrieve the items associated with *itemnums* 11 and 2.
- If the process was created with the CREATE intrinsic, retrieve the contents of the *param* parameter.

## Related Information

Intrinsics          None

Commands          None

Manuals            *Process Management Programmer's Guide (32650-90023)*

---

## GETJCW

NM and CM callable.

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

### Syntax

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

### Functional Return

*jcw*            **16-bit unsigned integer (assigned functional return)**

Returns the JCW. This word is structured by the calling program through the SETJCW or PUTJCW intrinsic.

### Related Information

Intrinsics      FINDJCW, PUTJCW, SETJCW

Commands       SETJCW, SHOWJCW

Manuals        *Interprocess Communication Programmer's Guide* (32650-90019)  
and *Command Interpreter Access and Variables Programmer's Guide*  
(32650-90011)

---

## GETLOCRIN

NM and CM callable.

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

### Syntax

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

### Parameters

*rincount*      16-bit unsigned integer by value (required)

Passes the number of local RINs to be acquired by the job/session. The maximum number of RINs available is defined when the system is configured.

### Operation Notes

All local RINs required by a job/session must be acquired in a single call to GETLOCRIN. If more local RINs are required, all local RINs must be released with the FREELOCRIN intrinsic before GETLOCRIN can return more.

### Condition Codes

- CCE (2)      Request granted.
- CCG (0)      Request denied. RINs already allocated to this job. Additional RINs cannot be allocated until these RINs are released.
- CCL (1)      Request denied. There are not enough RINs available for this call. RINs not allocated to this job.

### Related Information

- Intrinsics      None
- Commands      None
- Manuals      *Resource Management Programmer's Guide* (32650-90024)



---

## 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.

### Syntax

I16 <i>source</i> :=GETORIGIN;
-----------------------------------

### Functional Return

*source*      **16-bit signed integer (assigned functional return)**

Returns one of the following values:

Value	Meaning
0	Was not activated by either the parent or a child process
1	Activated by the parent process
2	Activated by a child process

### Related Information

Intrinsics      None

Commands      None

Manuals      *Process Management Programmer's Guide (32650-90023)*

---

## GETPRIORITY

NM and CM callable.

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

### Syntax

I16V    U16V    I16V GETPRIORITY( <i>pin</i> , <i>priorityclass</i> , <i>rank</i> );
---

### Parameters

***pin***            16-bit signed integer by value (required)

Passes the process whose priority is to be changed. To specify the calling process, set ***pin*** to zero. To specify a child process, set ***pin*** to the process identification number (PIN) of a child process.

***priorityclass*** 16-bit unsigned integer by value (required)

Contains the priority class in which the new process is scheduled (AS, BS, CS, DS, ES), computed in the following manner:

(ASCII of first character \* 256) +  
(ASCII of second character)

The integer equivalents are:

AS = 16,723  
BS = 16,979  
CS = 17,235  
DS = 17,491  
ES = 17,747

Aborts the calling process if ***priorityclass*** exceeds the maximum allowable priority class of the rescheduled process or specifies an invalid priority class.

Specify any priority class except AS, if in user mode; this function is limited by the maximum priority assigned to the account by the system manager. If in privileged mode, processes can be scheduled into all subqueues including AS. Processes in the linear queues, AS or BS, do not give up the CPU voluntarily; it could loop indefinitely and prevent other processes from accessing the CPU.

If in privileged mode, specify the *priorityclass* parameter as an absolute priority by *xA*, where *x* is an 8-bit priority number and *A* is the ASCII character *A*. For example, to request a priority class of 31 in the master queue, set *priorityclass* to %017501. An absolute priority must be specified in order to overcome the MAXPRI setting of an account.

Default: The priority class of the calling process

---

**Note** Scheduling a process into the AS or BS priority class can result in the rescheduled process deadlocking the system or locking out system and user processes from execution.

---

*rank*            **16-bit signed integer by value (optional)**  
 Used for backward-compatibility with pre-MPE IV operating systems only.

### Operation Notes

A process must be running in privileged mode to specify absolute priority.

A process can change its own priority or that of its child, but it cannot reschedule its parent.

### Condition Codes

CCE (2)        Request granted.  
 CCG (0)        Request denied. The process specified is not currently accessible (for example, unborn or dying).  
 CCL (1)        Request denied. An illegal or invalid PIN was specified.

### Related Information

Intrinsics     None  
 Commands     None  
 Manuals        *Process Management Programmer's Guide (32650-90023)*

---

## GETPRIVMODE

NM and CM callable.

Dynamically enters privileged mode. Privileged mode (PM) capability is required.

---

**Caution** Normal operating system checks and limitations are bypassed in privileged mode (PM). It is possible for a PM program to destroy file integrity, including the 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 operating system software.

---

### Syntax

```
GETPRIVMODE;
```

### Condition Codes

- CCE (2) Request granted. The program was in nonprivileged mode at the time of the intrinsic call.
- CCG (0) Request granted. The program was already in privileged mode at the time of the intrinsic call.
- CCL (1) Not returned.

### Operation Notes

After a successful call to GETPRIVMODE, the program remains in privileged mode until a call to GETUSERMODE or the program exits the current procedure. (A procedure refers to a Pascal procedure or function, a C function, a Fortran subprogram, an SPL procedure, or a COBOL program). Any procedures called while in privileged mode will execute in privileged mode. When the procedure that called GETPRIVMODE exits, the previous privilege level is restored.

### Related Information

Intrinsics	None
Commands	None
Manuals	None

## GETPROCID

NM and CM callable.

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

### Syntax

I16	I16V
<i>pin</i> := GETPROCID( <i>numchild</i> );	

### Functional Return

*pin*            **16-bit signed integer (assigned functional return)**

Returns the PIN of the specified child process.

### Parameters

*numchild*    **16-bit signed integer by value (required)**

Passes a number indicating which child's PIN GETPROCID returns. The value of *numchild* corresponds to the order of creation of the current children of the calling process, as follows:

Value	Meaning
$\leq 1$	PIN of the oldest child process
2	PIN of second oldest child process
.	.
.	.
.	.
<i>n</i>	PIN of the most recently created child process (where <i>n</i> is the number of children currently in existence)
$> n$	A value greater than the number of children in existence, returns a zero.

### Related Information

Intrinsics      None

Commands      None

Manuals        *Process Management Programmer's Guide (32650-90023)*

---

## GETPROCINFO

NM and CM callable.

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

### Syntax

I32	I16V
<i>processinfo</i> := GETPROCINFO( <i>pin</i> );	

### Functional Return

*processinfo*     **32-bit signed integer (assigned functional return)**

Returns the process management information about a parent or child process:

Bits	Value/Meaning
31:1	Activity state: 0    Process suspended 1    Process active
29:2	Suspension condition: If bit (31:1)=0: 00   Not used 01   Parent activation source 10   Child activation source 11   Either parent or child activation source
25:4	Reserved for the operating system
23:2	Origin of the last <b>ACTIVATE</b> intrinsic call: 00   operating system activation 01   Parent activation 10   Child activation 11   Not used
20:3	Queue characteristics: 001   DS or ES priority class 010   CS priority class 100   Linearly scheduled (AS, BS, or master queue)
16:4	Reserved for the operating system
8:8	Process priority number in master queue
0:8	Reserved for the operating system

## Parameters

*pin*            16-bit signed integer by value (required)

Passes the PIN of the process that the returned message pertains to. If requesting a parent process, set *pin* to zero. If requesting a child process, set *pin* to the process identification number (PIN) of that process.

## Condition Codes

CCE (2)        Request granted.

CCG (0)        Request denied. The process is already being terminated.

CCL (1)        Request denied. An illegal PIN was specified.

## Related Information

Intrinsics     None

Commands     None

Manuals       *Process Management Programmer's Guide (32650-90023)*

---

## GETUSERMODE

NM and CM callable.

Dynamically returns a program to nonprivileged mode.

### Syntax

```
GETUSERMODE;
```

### Condition Codes

- |         |   |
|---------|---|
| CCE (2) | Request granted. The program was in privileged mode at the time of the intrinsic call.    |
| CCG (0) | Request granted. The program was in nonprivileged mode at the time of the intrinsic call. |
| CCL (1) | Not returned.   |

### Related Information

- |            |      |
|------------|------|
| Intrinsics | None |
| Commands   | None |
| Manuals    | None |



## HP32208

CM callable only.

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

### Syntax

<p>D <i>version</i>:=HP32208</p>
--------------------------------------

### Functional Return

*version*      **32-bit signed integer (assigned functional return)**

Returns the VUF in the following format:

Bits	Value/Meaning
0:8	Version (ASCII format)
8:8	Update (binary format)
16:16	Fix level (binary format)

### Operation Notes

This intrinsic is for use with KSAM files only.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Using KSAM XL</i> (32650-90168)

---

## HPACDINFO

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

### Syntax

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

---

### Note

A maximum of four *itemnum2/item2* pairs can be specified. The last specified *itemnum2/item2* pair takes precedence over all identical *itemnum2/item2* pairs specified. Any conflict produces a warning message.

---

### Parameters

***status***      **32-bit signed integer by reference (required)**

Returns the status of the HPACDINFO call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:16) comprise *status.subsys*. The value represents the subsystem that sets the status information. The subsystem identifier for HPACDINFO is 246.

***itemnum1***      **integer by value (required)**

Passes the number indicating the *item*. Refer to Table 4-11.

***item1***      **Type varies (required)**

Passes the file or device identification information as specified in *itemnum*. Refer to Table 4-11.

*itemnum2*      **integer by value (optional)**

Passes the number indicating the *item2*. Refer to Table 4-12.

*item2*      **Type varies (optional)**

Returns the variable information as specified in *itemnum2*. Refer to Table 4-12.

Table 4-11. HPACDINFO Itemnum1/Item1 Values

Itemnum	Mnemonic	Item Description
The following constants have been defined for a file or device request:		
1	BA	Specifies the target file name. The <i>item1</i> value specified must be a byte array of 1 to 35 characters (for MPE filenames), word aligned. Note that no wildcard characters are allowed, a lockword can be included, and the file name specified does not need to be fully qualified and can be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
3	UDS	Specifies the unique file identifier (UFID). The <i>item1</i> value specified must be a 20-byte structure, which can be obtained from the file system through the <b>HPFOPEN</b> , <b>FLABELINFO</b> , or <b>FFILEINFO</b> intrinsics.
4	I	Specifies the file identifier of a previously opened file. The <i>item1</i> value specified must be a 16-bit integer value.
6	I	Specifies the logical device number (ldev). The <i>item1</i> value specified must be a 16-bit integer value.
7	BA	Specifies the name of a specific device. The <i>item1</i> value specified must be a 1 to 8 character byte array.
1	BA	Specifies the target file name. The <i>item1</i> value specified must be a byte array of 1 to 35 characters (for MPE filenames), word aligned. Note that no wildcard characters are allowed, a lockword can be included, and the file name specified does not need to be fully qualified and can be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
3	UDS	Specifies the unique file identifier (UFID). The <i>item1</i> value specified must be a 20-byte structure, which can be obtained from the file system through the <b>HPFOPEN</b> , <b>FLABELINFO</b> , or <b>FFILEINFO</b> intrinsics.
4	I	Specifies the file identifier of a previously opened file or directory. The <i>item1</i> value specified must be a 16-bit integer value.
6	I	Specifies the logical device number (ldev). The <i>item1</i> value specified must be a 16-bit integer value.
7	BA	Specifies the name of a specific device. The <i>item1</i> value specified must be a 1 to 8 character byte array.
15	BA	Specifies the target filename. The <i>item1</i> value specified must be a byte array of < 1024 characters long including the terminating carriage return, blank, or null character.

Table 4-12. HPACDINFO Itemnum2/Item2 Values

Itemnum	Mnemonic	Item Description
Security information that is not user-dependent can be requested using the following <i>itemnum2/item2</i> pairs:		
20	A	Returns a unique identifier representing the current version of the ACD. Currently the version number is 5. The <i>item2</i> value returned is a 16-bit integer.
21	I	Returns the number of ACD entries located in the target file ACD. The <i>item2</i> value returned is a 16-bit integer.
22	BA	Returns a unique identifier representing the first user specified in the ACD. The <i>item</i> value returned is a 1 to 18 character byte array.

Table 4-12. HPACDINFO Itemnum2/Item2 Values (continued)

Itemnum	Mnemonic	Item Description
Security information that is user-dependent can be requested using the following <i>itemnum2/item2</i> pairs:		
30	BA	Specifies the user whose ACD information is to be returned. The <i>item2</i> value specified must be an 18-character byte array, word aligned. If it is <18 characters, it must be terminated with a null character or a carriage return. This entry is required for subsequent user-dependent requests.
31	BA	Returns ACD modes in ASCII format. The <i>item2</i> value returned is a 25-character byte array.
32	I	Returns ACD modes as a bit mask. The <i>item2</i> value returned is a 16-bit integer whose bits have the following meaning:  <pre> r (0:1) - READ w (1:1) - WRITE x (2:1) - EXECUTE a (3:1) - APPEND l (4:1) - LOCK z (8:1) - Permission to read rd (11.1) - td (12.1) - cd (13.1) - dd (14.1) - n (15:1) - No access </pre>
33	BA	Returns ACD modes in ASCII format as access is granted the particular user. The <i>item2</i> value returned is a 25-character byte array. All wildcards are matched from the user specification. The access granted the user in the selected ACD pair can be greater than the access specified in the ACD pair if the user has AM or SM capability.
34	I	Returns ACD modes as a bit mask of all accesses granted the particular user. The <i>item2</i> value returned is a 16-bit integer whose bits have the following meanings:  <pre> r (0:1) - READ w (1:1) - WRITE x (2:1) - EXECUTE a (3:1) - APPEND l (4:1) - LOCK z (8:1) - Permission to read rd (11.1) - td (12.1) - cd (13.1) - dd (14.1) - n (15:1) - No access </pre> <p>All wildcards are matched from the user specification. The access granted the user in the selected ACD pair can be greater than the access specified in the ACD pair if the user has AM or SM capability.</p>
35	BA	Returns the next user in the ACD. The <i>item2</i> value returned is a 1 to 18 character byte array.

## Operation Notes

The HPACDINFO intrinsic lists the requested security attributes of a specified file or device. Three types of *itemnum/item* pair descriptors can be used to define supplied input or to specify returned ACD information. Such descriptors identify the file or device and request nonuser-dependent and user-dependent data.

## Related Information

Intrinsics        HPACDPUT, HPFOPEN

Commands        LISTF, LISTFILE, SHOWDEV

Manuals        *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364), *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042), *MPE/iX Utilities Manual* (32650-90081), and *Controlling System Activity* (32650-90155)

---

## 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);
```

### Parameters

***status***      **32-bit signed integer by reference (required)**

Returns the status of the HPACDPUT call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represents the subsystem that sets the status information. The subsystem identifier for HPACDPUT is 246.

***itemnum1***    **integer by value (required)**

Passes an identifier code defining the accompanying paired input (*item1*). The values for *itemnum1* are described in the *item1* parameter description. Refer to Table 4-13.

***item1***      **Type varies by reference (required)**

Passes the file or device identification information. The file or device identifier must be specified using one of the following predefined *itemnum1/item1* pairs. Refer to Table 4-13.

***itemnum2***    **Integer by value (required)**

Passes the function code defining the accompanying paired input (*item2*). The values for *itemnum2* are described in the *item2* parameter description. Refer to Table 4-14.



*item2*      **Type varies by reference (required)**

Passes the information to be used to manipulate the ACD of the specified file or device.

Functionality requests can be made using one of the predefined *itemnum2/item2* pairs. Refer to Table 4-14.

**Table 4-13. HPACDPUT Itemnum1/Item1 Values**

Itemnum	Mnemonic	Item Description
1	BA	Specifies the target file name. The <i>item1</i> value specified must be a byte array of 1 to 35 characters, word aligned.  No wildcard characters are allowed, a lockword can be included, the file name specified does not need to be fully qualified and can be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
2	BA	Specifies the target file name. The <i>item1</i> value specified must be a byte array of 1 to 35 characters, word aligned.  Wildcard characters are allowed, a lockword can be included, the file name specified does not need to be fully qualified, and a file name < 35 characters must be terminated with a null character or a carriage return.
3	UDS	Specifies the unique file identifier (UFID). The <i>item1</i> value specified must be a 20-byte structure, which can be obtained from the file system through the <b>HPFOPEN</b> , <b>FLABELINFO</b> , or <b>FFILEINFO</b> intrinsics.
4	I	Specifies the file number of a previously opened file. The <i>item1</i> value specified must be a 16-bit integer value.
5	N/A	Specifies all devices configured on the system. An <i>item1</i> value is not required with the option.
6	I	Specifies the logical device number (ldev). The <i>item1</i> value specified must be a 16-bit integer value.
7	BA	Specifies the name of a specific device. The <i>item1</i> value specified must be a 1 to 8 character byte array.
8	BA	Specifies the name of a specific device class. The <i>item1</i> value specified must be a 1 to 8 character byte array.
15	BA	Specifies the target filename. The <i>item1</i> value specified must be a byte array of < 1024 characters long, including the terminating carriage return, blank, or null character.

Table 4-14. HPACDPUT Itemnum2/Item2 Values

Itemnum	Mnemonic	Item Description
20	BA	Create a new ACD. The <i>item2</i> value specified must be a 1 to 279 character byte array.
21	BA	Add ACD pairs. The <i>item2</i> value specified must be a 1 to 279 character byte array.
22	BA	Replace ACD pairs. The <i>item2</i> value specified must be a 1 to 279 character byte array.
23	BA	Delete ACD pairs. The <i>item2</i> value specified must be a 1 to 279 character byte array.
24	N/A	Delete an ACD. An <i>item2</i> value is not required with this option.
25	BA	Create a new ACD. The <i>item2</i> value specified must be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
26	BA	Copy an ACD from a specified file. The <i>item2</i> value specified must be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
27	UDS	Copy an ACD from the specified unique file identifier (UFID). The <i>item2</i> value specified must be a 20-byte structure containing the UFID obtained from the file system through the <b>FLABELINFO</b> and <b>FFILEINFO</b> intrinsics.
28	I	Copy an ACD from a previously opened file number. The <i>item2</i> value specified must be a 16-bit integer containing the <i>filenum</i> of the file containing the ACD to be copied.
29	I	Copy an ACD from the specified logical device number (ldev). The <i>item2</i> value specified must be a 16-bit integer value.
30	BA	Copy an ACD from the specified device name. The <i>item2</i> value specified must be a 1 to 8 character byte array.
31	BA	Add ACD pairs. The <i>item2</i> value specified must be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
32	BA	Replace ACD pairs. The <i>item2</i> value specified must be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
33	BA	Delete ACD pairs. The <i>item2</i> value specified must be specified using an MPE or HFS name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
4-296	Command Definitions	The class indicating the maximum access to be calculated and assigned in the class mask. The <i>item2</i> value specified must be a 32-bit integer.

## Operation Notes

The HPACDPUT intrinsic allows programmatic adjustment, creation, and deletion of ACDs and their security provisions. The intrinsic is used to create, copy, and delete an ACD and to add, replace, modify, and delete pairs within an ACD.

## Related Information

Intrinsics	HPACDINFO, HPFOPEN
Commands	ALTSEC, COPY, PASSWORD, RELEASE, SECURE, STORE, RESTORE
Manuals	<i>MPE/iX Commands Reference Manual Volumes 1 and 2</i> (32650-90003 and 32650-90364), <i>System Startup, Configuration, and Shutdown Reference Manual</i> (32650-90042), <i>MPE/iX Utilities Manual</i> (32650-90081), and <i>Controlling System Activity</i> (32650-90155)

---

## HPCICOMMAND

NM callable only.

Executes a command programmatically.

### Syntax

CA	I16	I16	I16V
HPCICOMMAND( <i>cmdimage</i> , <i>cmderror</i> , <i>parmnum</i> , <i>msglevel</i> );			

### Parameters

*cmdimage*     **character array (required)**

Passes an ASCII string of <=279 characters consisting of a command and parameters, terminated by a carriage return. The carriage return character (%15) must be the last character of the command string. Do not include a prompt character in this string.

*cmderror*     **16-bit signed integer by reference (required)**

Returns any error code set by the last executed command. When the CI displays multiple errors for a single command, *cmderr* contains only the last error number displayed.

- If *cmderror* is zero, no error occurred.
- If *cmderror* is negative, a warning was detected.

This is the same error that would be returned by the command interpreter if *cmdimage* were executed interactively or in batch. The error message can be found in message set two of CATALOG.PUB.SYS.

Remote commands executed through the COMMAND or HPCICOMMAND intrinsics may not return a meaningful error status.

*parmnum*     **16-bit signed integer by reference (optional)**

Returns a number indicating the error:

- If *parmnum* is positive, it is a file system error number and *cmderror* is non-zero.
- If *parmnum* is negative, it is the column number where the error occurred and *cmderror* is non-zero. (The command interpreter (CI) prints a caret there.)
- If *parmnum* is zero, then no additional error-related data is available.

---

**Note** Check the *cmderror* parameter to verify if *cmdimage* succeeded. The *parmnum* may provide additional information.

---

*msglevel* **16-bit signed integer by value (optional)**

Passes one of the following values, indicating how to handle error and warning messages:

Value	Meaning
0	All error/warning messages will be printed to <code>\$STDLIST</code> (Default)
1	All CI error messages will be printed to <code>\$STDLIST</code> ; warning messages will be suppressed
2	No error/warning messages will be printed to <code>\$STDLIST</code> .

If a value other than 0, 1, or 2 is specified, *msglevel* defaults to 0 and CIWARN 9007 (Invalid message level - expected 0, 1, 2) is sent to `$STDLIST`.

HPCICOMMAND prints a caret if the CI does, when given the same *cmdimage* parameter and *msglevel*<2.

## Operation Notes

The HPCICOMMAND intrinsic allows the execution of UDCs and command files. RUN and other process creation commands are allowed if you or the program calling HPCICOMMAND have process handling (PH) capability.

The *cmdimage* parameter can contain any MPE/iX command except the following:

ABORT	JOB
BYE	LISTREDO
CHGROUP	MOUNT
DATA	MRJE
DISMOUNT	NRJE
DO	REDO
EOD	RESUME
EOJ	RJE
EXIT	SETCATALOG
HELLO	VSUSER

To control the error message display, use the *msglevel* parameter.

## Related Information

Intrinsics      `COMMAND`

Commands      None

Manuals      *Command Interpreter Access and Variables Programmer's Guide* (32650-90011) and *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364)

---

## HPCIDELETEVAR

NM callable only.

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

### Syntax

CA      I32 HPCIDELETEVAR( <i>varname</i> , <i>status</i> );
---

### Parameters

*varname*      **character array (required)**

Passes the name of a session-level variable whose value is to be retrieved. The name must begin with either an alphabetic character or the underscore character. It must consist of alphanumeric or underscore characters. The name is delimited by a nonalphanumeric, nonunderscore character, and can be up to 255 characters long. The name is not case-sensitive.

*status*      **32-bit signed integer by reference (optional)**

Returns the status of the HPCIDELETEVAR call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that sets the status information. The subsystem identifier for HPCIDELETEVAR is 166.

### Related Information

Intrinsics      HPCIPUTVAR, HPCIGETVAR

Commands      SETVAR, DELETEVAR, SHOWVAR

Manuals      *Command Interpreter Access and Variables Programmer's Guide* (32650-90011) and *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364)

---

## HPCIGETVAR

NM callable only.

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

### Syntax

CA      I32      U32V      * HPCIGETVAR( <i>varname</i> , <i>status</i> [, <i>itemnum</i> , <i>item</i> ] [ . . . ])
---

---

**Note**      You can specify up to six *itemnum/item* pairs.

---

### Parameters

*varname*      **character array (required)**

Passes the name of a session-level variable whose value is to be retrieved. The name must begin with either an alphabetic character or the underscore character. It must consist of alphanumeric or underscore characters. The name is delimited by a nonalphanumeric, nonunderscore character, and can be up to 255 characters long. The name is not case-sensitive.

*status*      **32-bit signed integer by reference (optional)**

Returns the status of the HPCIGETVAR call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:16) comprise *status.subsys*. The value represents the subsystem that set the status information. The subsystem identifier for HPCIGETVAR is 166.

## HPCIGETVAR

*itemnum*      **32-bit unsigned integer by value (optional)**  
Passes a number indicating the *item*, refer to Table 4-15.

*item*            **type varies (optional)**  
Returns variable information as specified in *itemnum*, refer to Table 4-15.

---

**Note**            If the type of variable is unknown, pass the maximum number of *itemnum, item* pairs (6) and check *itemnum* 13 (variable type) to determine which item holds the variable's value.

---



Table 4-15. HPCIGETVAR Itemnum/Item Values

Itemnum	Mnemonic	Item Description
0		<i>Itemnum/item</i> pair ignored
1	I32	Integer value of variable (output): 0 if variable is not an integer. Default: No value returned
2	CA	String value of variable (output): An ASCII 0 is returned if the value of the variable is not a string. Default: No value returned
3	I32	Boolean value of variable (output): 1 for a true value, 0 for a false value, 0 if variable is not a boolean. Default: No value returned
10	I32	Length of array passed to hold variable's string value (input): If a length is passed and an array is not, an error occurs. Default: 255
11	I32	Actual length (in bytes) of variable's string value (output) Default: No value returned
12	I32	Recursion used (input): Nonzero if the variable is to be recursively dereferenced; zero if the variable should not be recursively dereferenced (that is, it should retain its level one value). Default: 1 (nonzero)
13	I32	Type of variable ( <i>varname</i> , if found) (output) : The variable found returns the following: 1 Integer 2 String 3 Boolean Default: No value returned

---

**Note** For *itemnum*=12, recursively dereferenced variables are always returned as string values.

---

## **HPCIGETVAR**

### **Related Information**

Intrinsics      HPCIPUTVAR, HPCIDELETEVAR

Commands      DELETEVAR, SETVAR, SHOWVAR

Manuals      *Command Interpreter Access and Variables Programmer's Guide*  
(32650-90011) and *MPE/iX Commands Reference Manual Volumes 1 and 2*  
(32650-90003 and 32650-90364)

---

## HPCIPUTVAR

NM callable only.

Sets the value of a session-level variable.

### Syntax

CA	I32	U32V	*
HPCIPUTVAR( <i>varname</i> , <i>status</i> [, <i>itemnum</i> , <i>item</i> ] [...])			

---

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

---

### Parameters

*varname*      **character array (required)**

Passes the name of a session-level variable for which the value needs to be set. This name must begin with either an alphabetic character or the underscore character and consists of alphanumeric or underscore characters. The name is delimited by a nonalphanumeric, nonunderscore character, and must be <= 255 characters long. The name is not case-sensitive.

---

**Note** It is *not* a good practice to define CI variables with the same names as JCW's (job control words). Defining CI variables with the same names as JCW's may cause unpredictable results.

---

*status*      **32-bit signed integer by reference (required)**

Returns the status of the HPCIPUTVAR call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represents the subsystem that set the status information. The subsystem identifier for HPCIPUTVAR is 166.

## HPCIPUTVAR

*itemnum*      **32-bit unsigned integer by value (optional)**

Passes an item number indicating the *item*. Refer to Table 4-16.

*item*            **type varies by reference (optional)**

Passes the variable information specified in *itemnum*. Refer to Table 4-16.

**Table 4-16. HPCIPUTVAR Itemnum/Item Values**

Itemnum	Mnemonic	Item Description
0		<i>Itemnum/item</i> pair ignored
1	I32	Integer value assigned to specified variable name (input): No other pairs are needed and, if specified, will cause an error. Default: Not considered an integer value.
2	CA	String value assigned to specified variable name (input): Default: Not a string variables. Note: Requires itemnum 11. Itemnum 14 may be used with strings to set the type and value of <i>varname</i> .
3	I32	The Boolean value to be assigned to the specified variable name (input). Nonzero for a true value, zero for a false value. Default: Not a boolean variable.
11	I32V	Actual length of value string assigned to variable name (input): <i>Itemnum/item</i> pair 11 must be passed whenever <i>itemnum2</i> is passed.
14	I32	String interpretation (strings only; may be used only with <i>itemnum=2</i> ) (input). If you specify <i>itemnum=14</i> , the value of its corresponding <i>item</i> defines the type-representation (and the value) of <i>varname</i> based upon an interpretation of the <i>item</i> paired with <i>itemnum=2</i> . Refer to “Itemnum 14”. Default: 0 (zero).

### Itemnum 14

Given the (Pascal) code fragment

```
...
my_var := 'TESTVAR ';
var_value := 'Hello world';
val_len := 11;
item := 0; { default }
...
HPCIPUTVAR(my_var, status, 2, var_value,
           11, val_len, 14, item)
```

The value of *item* will set the type (and value) of *my\_var* based upon an interpretation of *var\_value*:

HPCIPUTVAR *itemnum* 14 with *item* values

Value of item	Value of var_value	Type/Value of my_var
=0	"TRUE"	Boolean (TRUE)
	"FALSE"	Boolean (FALSE)
	"123"	Integer (123)
	"other chars"	String ("other chars")
<>0	"TRUE"	String ("TRUE")
	"FALSE"	String ("FALSE")
	"123"	String ("123")
	"other chars"	String ("other chars")

**Operation Notes**

Allows the value of a session-level variable to be set. The name must be a valid MPE/iX variable name. The variable's value can be either a 32-bit signed integer, an array of characters, or a boolean (TRUE/FALSE).

**Related Information**

Intrinsics	HPCIGETVAR, HPCIDELETEVAR
Commands	SETVAR, DELETEVAR, SHOWVAR
Manuals	<i>Command Interpreter Access and Variables Programmer's Guide</i> (32650-90011) and <i>MPE/iX Commands Reference Manual Volumes 1 and 2</i> (32650-90003 and 32650-90364)

---

## HPDEBUG

NM callable only.

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

### Syntax

I32    CA        I32V    *
HPDEBUG( <i>status</i> , <i>cmdstr</i> [, <i>itemnum</i> , <i>item</i> ] [ . . . ] );

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPDEBUG call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (6:16) comprise *status.subsys*. The value represents the subsystem that sets the status information. The subsystem identifier for HPDEBUG is 165.

*cmdstr*

**character array (optional)**

Passes an array of <=1024 characters containing the system debugger commands. The first character in the buffer is recognized as the delimiter. The last character in the command string must be immediately followed by that same delimiter.

This command string is executed when the system debugger is called:

- For processes in jobs, process execution is resumed when the command string is exhausted.
- For processes in sessions, control remains in the debugger unless *cmdstr* contains a CONTINUE command.

- itemnum*      **32-bit signed integer by value (optional)**  
 Passes an item number indicating the *item*, refer to Table 4-17.
- item*            **type varies by value (optional)**  
 Passes the information specified in *itemnum*, refer to Table 4-17.

**Table 4-17. HPDEBUG Itemnum/Item Values**

Itemnum	Mnemonic	Item Description
1	I32	Output file number: Passes an opened file number where all output generated by Debug is sent. It must be an ASCII file with write access. The value 1 is valid and specifies that <b>\$STDLIST</b> is used. Default: Terminal LDEV for sessions and <b>\$STDLIST</b> for jobs.
2	I32	Welcome banner flag: Passes a value indicating whether the Debug welcome banner is printed or not. A value of 0 indicates do not print the welcome banner. Any other value causes the welcome banner to be displayed. Default: Print welcome banner (1)

## Operation Notes

The HPDEBUG intrinsic calls the system debugger with an optional character array containing Debug commands. This option distinguishes HPDEBUG from the DEBUG intrinsic. If the command list is defined, the debugger pushes the commands onto its command stack and executes them. If no command in the command string causes control to be returned to the calling procedure (for example, a CONTINUE command), the user is left in the debugger as long as the process is being run from a session environment. Processes run from a job are not allowed to stop in the system debugger. If the command string does cause control to return to the calling procedure, any remaining commands are left pending on the debugger's command stack to be executed the next time the debugger is called.

## Related Information

- Intrinsics      DEBUG, HPSETDUMP, STACKDUMP
- Commands      DEBUG
- Manuals        *MPE/iX System Debug Reference Manual (32650-90013)*

---

## 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.

### Syntax

I32   CA   I32   I32
HPDEVCONTROL( <i>status</i> , <i>ldev</i> , <i>itemnum</i> , <i>item</i> );

### Parameters

*status*      **32-bit integer (required)**

Returns the status of the HPDEVCONTROL call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represents the subsystem that sets the status information. There is no specific subsystem identifier for HPDEVCONTROL.

*ldev*          **character array (required)**

Passes the LDEV number of the device to be accessed. This parameter is a 200-character packed array.

---

**Note**          This parameter *must* be delimited with a printable, non-numeric character and *must* be zero left-filled:

"00000007" or  
a00000007a

Whatever delimiter *begins* the parameter must also *end* the parameter. The delimiter itself must not appear within the parameter.

a0000a007a will cause an error because a appears within the parameter.

---

**Note**          The LDEV parameter must be defined as 200 characters in the calling program. However, because logical device numbers are currently limited in size to 10 bytes only the *leftmost 10 bytes* of the array is used. For this reason, the *ldev* should be loaded into the leftmost 10 bytes of the 200-character array before the call is made.

---



It also may be important to understand that the system converts this parameter to a *pointer* before passing it to the intrinsic. It should *not* be coded as a pointer in the calling program, however.

**itemnum**      **32-bit integer (required)**

Specifies which operation is to be performed (refer to Table 4-18).

**item**            **32-bit integer (required)**

**Table 4-18. HPDEVCONTROL Itemnum/Item Values**

Itemnum	Mnemonic	Item Description
100	I32	Load Media: Attempts to load the media (tape devices only).
101	I32	Online: Programmatically places the peripheral online (tape devices only).

This intrinsic is not supported for the following devices: 7974, 7976 and 7978A. If a call is made against one of these devices an error is returned. Only item number 100 is supported for the 7978B.

---

**Note**            A call with an item number 100 (load) to a device which is already loaded is acceptable (produces no harmful result). However a tape device would go offline (only an HPIB DAT drive would not go offline). It is recommended that this intrinsic be called with item 100 followed by 101 whenever it is used for a tape device.

---

### Related Information

Intrinsics      None

Commands      None

Manuals        None

---

## HPDEVCREATE

Provides an application interface to create a streams, fifo, and device link file.

### Syntax

```
HPDEVCREATE ( CA pathname, I32V path_syntax, I32V path_length, status,  
  
I32V  
[ itemnum, item ] [ ... ] );
```

### Parameters

*pathname*     **32-bit pointer to a packed array of characters (required)**

The name of the device file to be created. If the *path\_length* does not indicate the length of the *pathname*, the device file name is assumed to be terminated with a null.

*path\_syntax*     **32-bit integer by value (optional)**

Indicates the syntax of the name in the *pathname* parameter. The valid values are:

0	MPE Escaped Syntax
1	MPE Only Syntax
2	POSIX Syntax

A value other than those listed above results in an error. The default value is 0 (MPE escaped syntax). MPE escaped syntax allows a name to be expressed in either MPE or POSIX syntax.

*path\_length*     **32-bit integer by value (optional)**

Indicates the length of the name in the *pathname* parameter. A positive value is taken as the exact length of the name, excluding any terminating characters.

If an invalid character is found in the *pathname*, an error is returned. Passing a -1 value indicates that the length of the name is unknown. HPDEVCREATE parses the name in the *pathname* parameter until it finds either a terminating character, an invalid character, or the name is 1023 characters. Passing a value of zero indicates a null or empty name will result in an error. Passing a negative value other than -1 will result in an error.

Default is -1.

*status*

**32-bit integer by reference (optional)**

Returns an indication of the success or failure of the intrinsic call.

- If no errors or warnings occur, status returns 32-bits of zero.
- If errors or warnings occur, status is interpreted as two 16-bit fields:

**Bits (0:16)**      Comprise `status.info`. A negative value indicates an error condition, and a positive value indicates a warning condition.

**Bits (16:16)**      Comprise `status.subsys`. The value represents the subsystem that sets the status information. For HPDEVCREATE intrinsic, the subsystem is `File System (143)`.

If the *status* parameter is not specified and an error occurs during the processing of HPDEVCREATE call, the calling process aborts.

*keyword#*

**32-bit integer by value (optional)**

Specifies a keyword value used in combination with the corresponding **keyvalue** parameter. Specifies the type of device file options used to create a device file. Up to 32 **keyword/keyvalue** pairs may be specified in a call to HPDEVCREATE. The following table lists the defined **keyword/keyvalue** pairs. Specifying a keyword other than those below will result in a error.

- If duplicate keywords are specified in a call to HPDEVCREATE, the last keyvalue is used, and a warning is returned to the caller.
- If no keyword/keyvalue pairs are specified, an error is returned.

*keyvalue#*

**Type varies by reference (optional)**

The **keyvalue** parameter is matched with its corresponding keyword to pass information to HPDEVCREATE. If a keyword is specified and the keyvalue is not present, an error will occur.

**Table 4-19. HPDEVCREATE Keywords/Keyvalues**

Keyword	Type	Description
0	None	<p>End of option list.</p> <p>There is no corresponding keyvalue. Terminates the keyword/keyvalue list and is equivalent to not passing a keyword/keyvalue pair.</p>
1	I32	<p>Device File Category.</p> <p>This option allows you to create one of several types of device files. The following is a list of valid categories:</p> <p>1                   FIFO</p> <p>                      Specifying this category creates a FIFO file. You can specify the ACD option with this item number. Any other option will result in an error.</p> <p>2                   STREAMS</p> <p>                      Specifying this category creates a STREAMS file using the attributes specified in the major number (item 3), minor number (item 4), and the link name (item 5). If this category is specified, then the major or minor number options are required.</p> <p>3                   DEVICE LINK</p> <p>                      Specifying this category creates a device_link file using the LDEV number specified in item 2 (LDEV). If you specify this category, then item 2 is required.</p>
2	I32	<p>LDEV</p> <p>Specifying this option causes a logical symbolic link to a device created using the LDEV numbet provided in the keyvalue parameter. This option is only valid when the device file category 3 (device link) is specified. If other options are specified, an error is returned.</p>
3	I32	<p>Major Number.</p> <p>This option allows specifications of device files in a traditional UNIX manner of major and minor numbers. If the major number is specified, the minor number option must also be specified, or an error is returned. The following major numbers are valid:</p> <p>1-254               Used to indicate a streams device file.</p> <p>                      Specifying a major number value other than 1-254 will result in an error.</p>

## HPDEVCREATE Keywords/Keyvalues

Keyword	Type	Description
4	I32	<p>Minor Number.</p> <p>This option is used in conjunction with option 3 to provide UNIX compatible device specification. If the minor number is specified, the major number must be specified, or an error is returned. The interpretation of the minor number is dependent on the value of the major number. The minor number represents a streams connection to a driver.</p> <p>A minor number is valid in the range from 0 to 16777215 (represented by 24 bits).</p>
5	CA	<p>Link Name.</p> <p>This option may be used with the major and minor number options (3 and 4) when a streams device is being created. An error is returned if this option is specified with any device file category other than <b>STREAMS</b>.</p> <p>The maximum size of this option is 8 characters (from the NMMGR link screen). A character placed in the first element designates the delimiter used to search to the end of the character array. For example:</p> <p><b>%linkname% (% is the delimiter)</b></p>
6	BA	<p>Access Control Definition (ACD):</p> <p>Passes a byte array defining the access control definition (ACD) to be attached to the streams or FIFO file. The array has a length of 1 to 279 bytes. This option expects a trailing carriage return character as a delimiter. For example:</p> <p><b>(X:@.@;R,W:JOE.SYS;RACD:SUE.SMITH)&lt;cr&gt;</b></p> <p>The &lt;cr&gt; is the carriage return character (13,0x0D).</p> <p>An error is returned if this item is specified when creating a device_link file.</p>

**Operation Notes**

When creating a device link, you are creating a symbol to a device with its default configuration parameters.

**Related Information**

Intrinsics	None
Commands	None
Manuals	None

---

## HPENBLTRAP

NM callable only.

Selectively enables or disables arithmetic traps.

### Syntax

I32V    I32 HPENBLTRAP( <i>mask</i> , <i>oldmask</i> );
--

### Parameters

*mask*      **32-bit signed integer by value (required)**

Passes a value indicating which arithmetic traps are enabled and which are not:

- If a bit is on (=1), the corresponding trap is enabled.
- If a bit is off (=0), the corresponding trap is disabled.

The bits and their associated arithmetic errors are:

Bits	Value/Meaning
31:1	3000 mode floating-point divide by zero
30:1	Integer divide by zero
29:1	3000 mode floating-point underflow
28:1	3000 mode floating-point overflow
27:1	Integer overflow
26:1	3000 mode double-precision overflow
25:1	3000 mode double-precision underflow
24:1	3000 mode double-precision divide by zero
23:1	Decimal overflow
22:1	Invalid ASCII digit
21:1	Invalid decimal digit
19:2	Reserved for the operating system
18:1	Decimal divide by zero
17:1	IEEE floating-point, inexact result
16:1	IEEE floating-point underflow

Bits	Value/Meaning
15:1	IEEE floating-point overflow
14:1	IEEE floating-point divide by zero
13:1	IEEE floating-point, invalid operation
12:1	Range errors
11:1	Software-detected NIL pointer reference
10:1	Software-detected misaligned result of pointer arithmetic or error in conversion from long pointer to short pointer
9:1	Unimplemented condition traps
8:1	Paragraph stack overflow
7:1	3000 mode packed decimal error
1:7	Reserved for MPE/iX
0:1	Assertion trap

**Note**

The following apply to various trap conditions represented in the *mask* parameter:

- Native mode supports two floating-point formats: IEEE and 3000 mode. Both execute in native mode, but 3000 mode performs HP 3000 type manipulations. Since it is possible to use both formats during program execution, there are separate bits in the *mask* for enabling/disabling traps of these formats.
- Some error conditions specified are not strictly arithmetic traps (for example, range errors, nil pointers, and paragraph stack overflow). However, many arithmetic traps are caught by reserved instructions that raise the conditional traps. For this reason, all are enabled/disabled by HPENBLTRAP.
- Some of the instructions that raise conditional traps are reserved to indicate some of the above trap conditions. A nonreserved instruction is one not generated by a compiler. If a nonreserved instruction causes a conditional trap, this is reported as an unimplemented condition trap.

*oldmask*

**32-bit signed integer by reference (required)**

Returns the value of the previous *mask* to the program.

## HPENBLTRAP

### Operation Notes

Allows selective enabling or disabling of arithmetic traps. It provides more flexibility than the ARITRAP intrinsic, which collectively enables or disables traps.

There is a difference between arming and enabling traps:

- Enabling a trap means that the occurrence of a trap condition is not ignored.
- Arming a trap is required so that, on a trap condition, a user-written routine is invoked and can take appropriate recovery actions.

The following list summarizes what can occur when an arithmetic trap condition arises:

- If a trap is both enabled and armed, the user-written trap handler is invoked whenever a trap condition occurs.
- If a trap is enabled but not armed, one of two situations applies:
  - If a Pascal/XL TRY statement has been executed, control is passed to the recover block by doing an escape.
  - If a Pascal/XL TRY statement has not been executed, an error message is output and the process aborts.
- If a trap is disabled, irrespective of whether it is armed, the trap is ignored, and execution of the process continues without any interruption.

---

**Note** By default, all traps except IEEE floating-point exceptions are enabled, and the system trap handler is armed. Many floating-point operations result in an inexact result. Consequently, most compiler libraries doing floating-point operations result in an inexact trap if the IEEE inexact result trap is enabled.

---

### Condition Codes

CCE (2) Request granted. All traps were originally disabled.  
CCG (0) Request granted. At least one trap was originally enabled.  
CCL (1) Not returned.

### Related Information

Intrinsics None  
Commands None  
Manuals *Trap Handling Programmer's Guide* (32650-90026)



---

## HPERRDEPTH

NM callable only.

Returns the current depth of the process error stack.

### Syntax

I32    I32 HPERRDEPTH( <i>depth</i> , <i>status</i> );
---

### Parameters

*depth*    **32-bit signed integer by reference (required)**

Returns the current number of entries on the process error stack.

*status*    **32-bit signed integer by reference (optional)**

Returns the status of the HPERRDEPTH call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represents the subsystem that set the status information. The subsystem identifier for HPERRDEPTH is 187.

Default = NIL

---

**Caution**    If an error or warning is encountered and the *status* parameter was not specified, HPERRDEPTH causes the calling process to abort.

---

### Operation Notes

HPERRDEPTH can be used to check the depth of the error stack before using HPERRMSG.

---

**Note**    With the exception of HPERRMSG, HPERRREAD, and HPERRDEPTH, all intrinsics that are called clear the process error stack.

If other intrinsics are called before or during the process of reading the error stack, then the stack is cleared and the information it contained is lost.

---

## HPERRDEPTH

### Related Information

Intrinsics      HPERRMSG, HPERRREAD''

Commands      None

Manuals      *Getting System Information Programmer's Guide* (32650-90018) and *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368)

## HPERRMSG

NM callable only.

Obtains or displays error messages from the system catalog.

### Syntax

```

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

### Parameters

*displaycode*    **32-bit signed integer by value (required)**

Specifies the operation to be performed:

Value	Meaning
1	Dump the error stack to <b>\$STDLIST</b> and display any errors that are detected by <b>HPERRMSG</b> (returned in <i>status</i> ).
2	Display to <b>\$STDLIST</b> the error message corresponding to the value in <i>errornum</i> and display any errors that are detected by <b>HPERRMSG</b> (returned in <i>status</i> ).
3	Obtain the error message corresponding to the value in <i>errornum</i> and place it in the buffer specified by <i>buffer</i> .
4	Display to <b>\$STDLIST</b> the number of stack errors specified in <i>depth</i> and display any errors detected by <b>HPERRMSG</b> (returned in <i>status</i> ).
5	Display to <b>\$STDLIST</b> the error message corresponding to <i>errornum</i> . Do not display any errors that are detected by <b>HPERRMSG</b> (returned in <i>status</i> ), but save information in the <i>status</i> parameter.
6	Obtain the subsystem/error identification number corresponding to the error number in <i>errornum</i> , and place it in the buffer specified by <i>buffer</i> .

## HPERRMSG

Value	Meaning
7	Get messages for the number of stack errors specified in <i>depth</i> and place them in the buffer specified by <i>buffer</i> . The contents of <i>buffer</i> contains the same information as if the user had specified <i>displaycode</i> 4, including errors that are detected by <b>HPERRMSG</b> . All messages are separated by carriage return/line feed characters.
8	Dump the error stack to the buffer specified by <i>buffer</i> . The result is the same as with <i>displaycode</i> 1, except that the messages are placed in <i>buffer</i> , separated by carriage return/line feed characters.

*depth*      **32-bit signed integer by value (optional)**

Specifies the number of error entries from the stack that are to be displayed. A 1 (one) implies the last error entry made.

Default = 0

*errorproc*      **16-bit signed integer (reserved)**

Do not specify this parameter, but maintain parameter position.

Default = 0

*errornum*      **32-bit signed integer by value (optional)**

Passes a subsystem error identification number whose corresponding message from the system catalog is to be displayed or returned. An *errornum* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information.

If not specified, defaults to 0.

*buffer*      **character array (optional)**

Specifies the buffer where text from the system catalog error messages are to be placed.

If not specified, defaults to NIL.

*buflength***16-bit signed integer by reference (optional)**

Passes the length of the buffer input:

- As an input parameter, the length of the buffer where the error messages are to be placed. If the resulting messages are longer than this length, they are truncated. This procedure does not verify that the *buffer* parameter has enough storage to hold the number of bytes specified by *buflength*. If there is not enough storage, the caller's stack may be corrupted.
- As an output parameter, the actual length (in bytes) of the messages that were returned.

If *buffer* is specified, *buflength* must also be specified.

If not specified, defaults to NIL.

*status***32-bit signed integer by reference (optional)**

Returns the status of the HPERRMSG call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *errornum* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPERRMSG is 187.

If not specified, defaults to NIL.

---

**Caution**

If an error or warning is encountered and the *status* parameter was not specified, HPERRMSG causes the calling process to abort.

---

## HPERRMSG

### Operation Notes

Allows the user to obtain or display error messages from the system catalog by supplying the corresponding error statuses. The caller can supply a status whose corresponding message is to be displayed, or specify that the messages corresponding to one or more errors in the process error stack be displayed. The caller can optionally provide a buffer and ask that the error messages be placed in that buffer rather than be displayed to the user.

For errors that are displayed from the process error stack, any parameters that were pushed with the errors are inserted into the corresponding messages. Also, when dumping the entire stack, the user is warned if any entries on the process stack were overwritten. (This happens if more than 16 entries were pushed onto the stack.) If any entries were overwritten, the oldest entries are the ones lost.

---

**Note** With the exception of HPERRMSG, HPERRREAD, and HPERRDEPTH, all intrinsics that are called clear the process error stack.

If other intrinsics are called before or during the process of reading the error stack, then the stack is cleared and the information it contained is lost.

---

**Note** The internal software which moves error codes onto the error stack is a proprietary part of the MPE/iX operating system. Hewlett-Packard currently provides no user-callable procedure to move error codes onto the error stack.

---

### Related Information

Intrinsics      HPERRDEPTH, HPERRREAD

Commands      None

Manuals      *Getting System Information Programmer's Guide* (32650-90018) and *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368)

---

## HPERRREAD

NM callable only.

Reads any specified error from the process stack.

### Syntax

I32V	I32	I32	I32
HPERRREAD( <i>depth,errornum,procnum,status</i> )			

### Parameters

- depth*      **32-bit signed integer by value (required)**
- Passes the index of the stack entry to read from. A 1 (one) implies the last entry that was pushed. Specification of depth equal to that of the actual stack depth implies the oldest entry on the stack. An error is returned if the value of this parameter does not correspond to an error in the stack.
- errornum*    **32-bit signed integer by reference (required)**
- Returns the error status that was read from the stack. This value is only valid if *status* is zero.
- Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition.
- Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information.
- procnum*      **32-bit signed integer by reference (optional)**
- Do not specify this parameter. Instead, maintain the parameter position with a comma. Always returns zero.
- If not specified, defaults to NIL.
- status*        **32-bit signed integer by reference (optional)**
- Returns the status of the HPERRREAD call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *errornum* is interpreted as two 16-bit fields.

## HPERRREAD

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:16) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPERRREAD is 187.

If not specified, defaults to NIL.

---

**Caution** If an error or warning is encountered and the *status* parameter was not specified, HPERRREAD causes the calling process to abort.

---

### Operation Notes

Allows the user to read any specified error from the process stack. The user specifies how far down into the stack to read, and the corresponding error status is returned in *errornum*. If any parameters were pushed onto the stack with the specified error, this intrinsic does not return them. Refer to the HPERRMSG intrinsic for information on how to display errors with their parameters.

---

**Note** With the exception of HPERRMSG, HPERRREAD, and HPERRDEPTH, all intrinsics that are called clear the process error stack.

If other intrinsics are called before or during the process of reading the error stack, the stack is cleared, and the information it contained is lost.

---

**Note** The internal software which moves error codes onto the error stack is a proprietary part of the MPE/iX operating system. Hewlett-Packard currently provides no user-callable procedure to move error codes onto the error stack.

---

### Related Information

Intrinsics      HPERRDEPTH, HPERRMSG

Commands        None

Manuals        *Getting System Information Programmer's Guide* (32650-90018) and *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368)



## HPFDUPLICATE

NM callable only.

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

### Syntax

I16	I32	I32V	*
<i>filenum</i> := HPFDUPLICATE( <i>source</i> , <i>status</i> , <i>target</i> );			

### Functional Return

*filenum*      **16-bit signed integer** (assigned functional return)

Returns the actual file number used to duplicate the *source* file. This file number can be different than the file number requested in the *target* parameter. If an error occurs during the duplication process, a -1 value is returned.

### Parameters

*source*      **32-bit signed integer by value (required)**

Specifies the file number to be duplicated. The file number must be a valid, open file.

*status*      **32-bit signed integer by reference (optional)**

Returns the status of the HPFDUPLICATE call. If an error occurs and this parameter was not specified, the functional return can be used to determine the success or failure of the call.

*target*      **32-bit signed integer by reference (optional)**

Specifies the file number to be duplicated in the *source* parameter. If the file number is already opened, the next highest available file number is used. The value passed must be within the legal range of file numbers. The default value for this parameter is zero (0).

### Operation Notes

When a file descriptor is duplicated, the duplicate file descriptor shares the same data pointer as the original file descriptor in the same way that file descriptors share the same data pointer with MULTI access. This gives you the ability to perform operations on files that have not been previously available. Specifically, files that are opened for SHARED or EXCLUSIVE access can now have their data pointers shared among multiple file descriptors, at the discretion of the application that originally opens the file descriptor.

## HPFDUPLICATE

### Related Information

Intrinsics	FOPEN, HPFOPEN
Commands	None
Manuals	None

## HPFIRSTLIBRARY

NM callable only.

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

### Syntax

```

CA      I32  I32  CA
HPFIRSTLIBRARY(formaldesig, status, length, firstlib,
               I32      I32
               firstlib_length, firstlib_syntax);

```

### Parameters

***formaldesig*** character array (required)

Returns the fully qualified MPE syntax file name of the first XL in the binding sequence of the calling process.

The *formaldesig* parameter must be at least 28 bytes in length in order to contain the longest possible MPE syntax file name with delimiters. The lockword is not returned. The first and last characters of the returned value are blanks that act as delimiters.

***status*** 32-bit signed integer by reference (optional)

HPFIRSTLIBRARY returns a NM Loader error 128 when the first library name cannot be represented using the syntax that the user specifies.

***length*** 32-bit signed integer by reference (optional)

Returns the length of *formaldesig* parameter. The length includes the surrounding delimiters.

***firstlib*** character array (optional)

Returns the absolute pathname or fully qualified MPE syntax name of the first executable library in the binding sequence of the calling process.

The *firstlib* parameter contains the name with delimiters. The first and last characters of the returned value are blanks that act as delimiters.

***firstlib\_length*** 32-bit signed integer by reference (optional)

On input, the *firstlib\_length* parameter specifies the maximum *firstlib* length in bytes.

On output, the *firstlib\_length* parameter returns the actual length of *firstlib*. The length includes the surrounding delimiters.

If *firstlib\_length* is incorrect when input, variables allocated near *firstlib* can be overwritten, or an error could occur. If the *firstlib\_length* parameter is not

## HPFIRSTLIBRARY

specified, HPFIRSTLIBRARY determines if the *firstlib* parameter is long enough to hold the first library name to be returned.

Note that a value is not returned in *firstlib\_length* if the *firstlib* parameter is not specified.

*firstlib\_syntax* **32-bit signed integer by reference (optional)**

On input, the *firstlib\_syntax* parameter specifies the syntax that will be used when HPFIRSTLIBRARY returns a name in the *firstlib* parameter.

On output, the *firstlib\_syntax* parameter returns the syntax that is actually used. This is only useful when MPE-escaped syntax is specified on input and the name can legally be either an MPE syntax name or an HFS syntax name.

Note that a value is not returned in *firstlib\_syntax* if the *firstlib* parameter is not specified. The possible syntaxes are MPE-escaped syntax (0), MPE-only syntax (1), and HFS syntax (2)/.

Default is MPE-only syntax.

### Operation Notes

This intrinsic searches files in the binding sequence for a procedure, beginning with the first XL. The first XL is the second file in the binding sequence (located immediately after the program file).

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Resource Management Programmer's Guide</i> (32650-90024)

---

## HPFOPEN

NM callable only.

Creates file objects and supports FIFO files.

### Syntax

I32      I32      I32V    * HPFOPEN( <i>filenum</i> , <i>status</i> [, <i>itemnum</i> , <i>item</i> ] [...]);
--

---

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

---

### Parameters

*filenum*        **32-bit signed integer by reference (required)**

Returns a file descriptor for a FIFO file.

Can be used safely with all file system intrinsics that require a 16-bit file number to be passed in the intrinsic call (for example, `FREAD`, `FWRITE`, `FCLOSE`).

*status*         **32-bit signed integer by reference (optional)**

Returns the status of the HPFOPEN call. If no errors or warnings are encountered, *status* returns 32-bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields:

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represents the subsystem that set the status information. The subsystem identifier for HPFOPEN is 143.

---

**Caution**        If an error or warning is encountered and the *status* parameter was not specified, HPFOPEN causes the calling process to abort.

---

## HPFOPEN

*itemnum*      **32-bit signed integer by value (optional)**

Passes the item number. Refer to Table 4-20.

*item*            **type varies by reference (optional)**

Passes and/or returns the option indicated by the corresponding *itemnum* parameter. Refer to Table 4-20.

---

**Note**

An *itemnum* takes precedence over any previously specified duplicate *itemnum*. Any duplicated *itemnum* is flagged as a warning.

---

Table 4-20. HPFOPEN Itemnum/Item Values

Itemnum	Mnemonic	Item Description
0		<p>End of option list:</p> <p>There is no corresponding <i>item</i>. The absence of an <i>itemnum</i> after the last <i>itemnum,item</i> pair is equivalent to specifying this option.</p>
2	CA	<p>Formal designator:</p> <p>Passes a formal file designator that is interpreted according to MPE-escaped semantics (unless another syntax has been chosen via item 41). The file name must be terminated by a nonalphanumeric character other than a period (<code>.</code>), a slash (<code>/</code>), a hyphen (<code>-</code>), and an underscore (<code>_</code>). Use of matched starting and ending name delimiters (a quoted name) alleviates the need for a terminating character other than the quote characters.</p> <p>The file referred to by <i>formaldesig</i> can be either an MPE file (i.e., one that uses MPE syntax) or it can follow HFS syntax. If <i>formaldesig</i> follows MPE syntax, the file name can include password, group, and account specifications. The file name can backreference a file equation and optionally be preceded by an asterisk. If <i>formaldesig</i> follows HFS syntax, the file name must start with either a dot (<code>.</code>) or a slash (<code>/</code>).</p> <p>The file referred to by <i>formaldesig</i> may reside either in an MPE group or in an HFS directory. For files located in HFS directories, traverse directory entries (TD) access is required to all directories specified in <i>formaldesig</i>. If there is no TD access, HPFOPEN fails and a value of <code>-180</code> is returned in the <i>status.info</i> parameter. If you are using HPFOPEN to create a file or hierarchical directory and you lack of create directory entry (CD) access, <i>status.info</i> will return a value of <code>-179</code>.</p> <p>If <i>formaldesig</i> is an escaped pathname:</p> <ul style="list-style-type: none"> <li>■ you cannot reference remote files</li> <li>■ it cannot express a name equivalent to <i>filename:envid</i></li> <li>■ you cannot use the <i>device</i> parameter (<i>device=node#</i>) to specify the remote location of a device</li> </ul> <p>If <i>formaldesig</i> is the name of a user-defined file, it can begin with an asterisk (<code>*</code>). If <i>formaldesig</i> is the name of a system-defined file, it can begin with a dollar sign (<code>\$</code>). When creating a KSAM file, <i>formaldesig</i> must be a unique file name, that is, one not currently existing in the permanent file directory.</p> <p>The formal file designator can contain command interpreter variables and expressions that are evaluated before <i>formaldesig</i> is parsed and validated.</p> <p>As the default, HPFOPEN creates a nameless file that can be read or written to, but not saved. (The domain option of a nameless file must specify a new file unless it is a device file.)</p>

## HPFOPEN

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
2 cont'd		<p>The following are examples of valid formal file designators:</p> <pre>&amp;file/lock.group.account:node.dest.level&amp; &amp;filename&amp; &amp;!myfile&amp; &amp;!afile/![FINFO("!afile",33)]&amp;</pre> <p>The following are examples of invalid formal file designators:</p> <pre>"filename.group (missing delimiter (")) file.group" ( 'f' is used as delimiter, missing at end)</pre> <p>(ASC) It is recommended that this <i>itemnum</i>, <i>item</i> pair be used for asynchronous devices.</p> <p>When you use <b>HPFOPEN</b> to open a file, you may use either <i>itemnum</i>=2 or <i>itemnum</i>=51; you cannot use both.</p>
3	I32	<p>Domain:</p> <p>Passes a value indicating which file domain the system searches to locate the file. A nameless disk file must always be a new file.</p> <p>A device file (such as a tape drive, terminal, spooled printer or <i>hot</i> printer) always resides in the system file domain (permanent file directory). Always specify a device file as old or permanent.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 The file is a new temporary file. It is not placed in a directory.</li> <li>1 The file is a permanent file, found in the system file domain.</li> <li>2 The file is a temporary file, found in the job file domain.</li> <li>3 The file is an old (permanent or temporary) file. The job file domain is searched first. If the file is not found, the system file domain is searched.</li> <li>4 The file is created, placed in the permanent file directory, and becomes a permanent file.</li> </ul> <p>Hierarchical directories must be created in the permanent file domain by specifying the create file domain (4).</p> <p>Default: 0</p>



Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
5	I32	<p>Designator:</p> <p>Passes a value indicating a special file opening. Any of the following special files can be specified with the <i>itemnum</i>=2. For example, a file name of <b>\$STDLIST</b> opens the standard list device. The following values are valid:</p> <ul style="list-style-type: none"> <li>0 Allows all other options to specify the file.</li> <li>1 The actual file designator is <b>\$STDLIST</b>.</li> <li>2 The actual file designator is <b>\$NEWPASS</b>.</li> <li>3 The actual file designator is <b>\$OLDPASS</b>.</li> <li>4 The actual file designator is <b>\$STDIN</b>.</li> <li>5 The actual file designator is <b>\$STDINX</b>.</li> <li>6 The actual file designator is <b>\$NULL</b>.</li> </ul> <p>Default: 0</p> <p>For example, if <b>MYFILE</b> is passed in <i>itemnum</i>=2, then using <i>itemnum</i>=5 and <i>item</i>=4 to equate it with <b>\$STDIN</b> is equivalent to the file equation <b>FILE MYFILE=\$STDIN</b>.</p> <p>This option is not equated with <i>itemnum</i>=2 if both of the following conditions are true:</p> <ul style="list-style-type: none"> <li>■ The <i>itemnum</i>=9 option allows file equations for the file opening.</li> <li>■ An explicit or implicit <b>FILE</b> command equating the formal file designator to a different actual file designator occurs in the job/session.</li> </ul> <p>A leading * in a formal file designator passed by <i>itemnum</i>=2 overrides an <i>itemnum</i>=9 option.</p>
6	I32	<p>Record format:</p> <p>Passes a value indicating the internal record structure desired for the file. This option is applicable only at file creation.</p> <p>The following values are valid for records:</p> <ul style="list-style-type: none"> <li>0 Fixed-length</li> <li>1 Variable-length</li> <li>2 Undefined-length (no implied structure)</li> <li>9 Byte stream</li> <li>10 Hierarchical directory</li> </ul> <p>Default: 0</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
6 cont'd		<p>Byte stream record format may be specified only for standard disk files (<i>itemnum</i> 10 equal to 0). Hierarchical directory record format is the the default record format when creatinng a directory (<i>itemnum</i> 10 equal to 9). <i>Itemnum</i> 10 equal to 9 is the only record format which may be specified when creating a directory. Hierarchical record format is only specified for the directory file type. Record formats not implemented for the specified file type are ignored. Byte stream and hierarchical directory record formats are supported only on disk devices.</p> <p>(ASC) This <i>itemnum,item</i> pair is ignored for files opened on a terminal; records of files on terminals are of undefined length. If the file is to be redirected to tape or disk, set the value to 0 (fixed-length).</p> <p>FIFO file must be created with the byte streams (9) record format.</p>
7	I32	<p>Carriage-control:</p> <p>Passes a value indicating whether or not a carriage-control directive is supplied in the calling sequence of each <b>FWRITE</b> call that writes records onto the file. This option is applicable only at file creation.</p> <p>The following values are valid:</p> <p>0 No carriage-control directive expected</p> <p>1 Carriage-control directive expected</p> <p>Default: 0</p> <p>Carriage-control is defined only for ASCII files. This option and <i>itemnum</i>=53 are exclusive, and attempts to open new files with both binary and carriage-control directives result in an access violation.</p> <p>A carriage-control character passed through the <i>control</i> parameter of <b>FWRITE</b> is recognized for files with carriage-control specified in <b>HPFOPEN/FOPEN</b>. Embedded control characters are treated as data on files where no carriage-control is specified, and spacing is not invoked for the file. Specify spacing action on files where carriage-control has been specified by either embedding the control in the record, indicated with a <i>control</i> parameter in the call to <b>FWRITE</b>, or by sending the control code directly through the <i>control</i> parameter of <b>FWRITE</b>.</p> <p>If a carriage-control character is sent to a file where the control cannot be executed directly (for example, line spacing characters sent to a disk or tape file), the control character is embedded as the first byte of the record. Therefore, the first byte of each record in a disk file having carriage-control characters enabled contains control information. If carriage-control characters are sent to other types of files, the control is transmitted to the driver.</p> <p>Control codes %400 through %403 are remapped to %100 through %103, so that they fit into one byte and can be embedded. Records written to the line printer with control codes %400 through %403 should contain only control information.</p> <p>Records written with control codes %400 through %403 and no data (count=0, or embedded control and count=1) does not cause physical I/O.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
7 cont'd		<p>For computing record size, the file system considers carriage-control information as part of the data record. Therefore, specifying the carriage-control option adds one byte to the record size when the file is originally created. For example, a specification of <code>REC=-132,1,F,ASCII;CCTL</code> results in a <i>resize</i> of 133 bytes.</p> <p>Generally, the entire record can be read. Refer to the table listing the item values returned by the <code>FFILEINFO</code> intrinsic. However, on writes to files where carriage-control characters are specified, the data transferred is limited to <i>resize</i>-1 unless a control of one is passed, indicating the data record is prefixed with embedded carriage-control characters.</p> <p>The value of this <i>itemnum</i> is ignored when a byte stream or hierarchical directory is created. Byte stream files and hierarchical directories are created without carriage control (<code>NOCCTL</code>).</p>
8	CA	<p>Enable tape label:</p> <p>Passes the tape label name of a labeled tape. The name must follow the ANSI standards for tape label names. The name consists of &lt;=6 printable characters that identify the volume. In a multivolume set, only the first tape label can be specified.</p> <p>Default: a null tape label</p> <p>A character placed in the first element designates the delimiter used by <code>HPFOPEN</code> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example:</p> <p style="padding-left: 40px;"><i>%valid%</i> (% is the delimiter, <i>valid</i> is the designator)  <i>fabxyzf</i> (<i>f</i> is the delimiter, <i>abxyz</i> is the designator)</p> <p>(ASC) Not used for asynchronous devices.</p>
9	I32	<p>Disallow file equation:</p> <p>Passes a value indicating whether or not file equations are allowed. A leading * in a formal file designator overrides the setting to disallow file equations.</p> <p>The following values are valid:</p> <p>0 Allow file equations to override programmatic or system-defined file specifications.</p> <p>1 Disallow file equations from overriding programmatic or system-defined file specifications.</p> <p>File equations can be enabled for escaped pathnames expressed using MPE-escaped name semantics or names expressed using POSIX name semantics, but a matching file equation is not found since the file designator on the left side of a file equation can only be expressed using MPE-only syntax.</p> <p>Default: 0</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
10	I32	<p>File type:</p> <p>Passes a value indicating the internal record structure used to access records in the file. If the file is old, this option is ignored. Specifying an <i>itemnum=5</i> value other than zero overrides this option. This option is applicable only at file creation.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 Standard (STD) file</li> <li>1 KSAM/3000 file</li> <li>2 Relative I/O (RIO) file</li> <li>3 KSAM XL file</li> <li>5 NM spoolfile</li> <li>4 Circular (CIR) file</li> <li>6 Message (MSG) file</li> <li>9 Directory</li> </ul> <p>Default: 0</p> <p>Hierarchical directories must be created in the permanent file domain.</p> <p>KSAM/3000 (1), RIO (2) and CIR (4) file types may only be created using names belonging to the MPE name space.</p> <p>(ASC) Set the value to 0 for asynchronous devices.</p>
11	I32	<p>Access type:</p> <p>Passes a value indicating the type of access intended for the file. This option restricts/allows usage of the file system intrinsics.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 Read access only, if the file's security provisions allow read access. <b>FWRITE</b>, <b>FUPDATE</b>, and <b>FWRITEDIR</b> intrinsic calls cannot reference this file. The end-of-file (EOF) is not changed; the record pointer starts at zero. (Default)</li> <li>1 Write access only, if the file's security provisions allow write access. Any data written in the file prior to the current <b>HPFOPEN</b> request is deleted. <b>FREAD</b>, <b>FREADSEEK</b>, <b>FUPDATE</b>, and <b>FREADDIR</b> intrinsic calls cannot reference this file. The EOF is set to zero; the record pointer starts at zero. On magnetic tape an EOF is written to the tape when the file is closed even if no data is written.</li> <li>2 Write-Save access only, if the file's security provisions allow write access. Previous data in the file is not deleted. <b>FREAD</b>, <b>FREADSEEK</b>, <b>FUPDATE</b>, and <b>FREADDIR</b> intrinsic calls cannot reference this file. The EOF is not changed; the record pointer starts at zero. Therefore, data is overwritten if <b>FWRITE</b> is called. The system changes this value to append for message files.</li> </ul>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
11 cont'd		3 Append access only, if the file's security provisions allow either append or write access. <b>FREAD</b> , <b>FREADDIR</b> , <b>FREADSEEK</b> , <b>FUPDATE</b> , <b>FSPACE</b> , <b>FPOINT</b> , and <b>FWRITEDIR</b> intrinsic calls cannot reference this file. The record pointer is set to EOF prior to each <b>FWRITE</b> . For disk files, the EOF is updated after each <b>FWRITE</b> call. Therefore, data cannot be overwritten.
		4 Read/Write (I/O) access only, if the file's security provisions allow both read and write access. If both read and write access are not allowed, the access type is limited to that specified in the security provisions (either read or write). Any file intrinsic can be issued except <b>FUPDATE</b> for this file. The EOF is not changed; the record pointer starts at zero. This option is not valid for message files.
		5 Update access only, if the file's security provisions allow both read and write access. If both read and write access are not allowed, the access type is limited to that specified in the security provisions (either read or write). All file intrinsics can be issued, including <b>FUPDATE</b> , for this file. The EOF is not changed; the record pointer starts at zero. This option is not valid for message files.
		6 Execute access only, if the file's security provisions allow execute access. This allows read/write access to any loaded file. The program must be running in privileged mode to specify execute access. This option is not valid for message files.
		7 Execute-Read access only, if the file's security provisions allow execute access. This allows only read access to a loaded file. The program must be running in PM to specify execute-read access. This is changed to execute access for KSAM, CIR, and RIO files. Not valid for message files.
		8 Reserved for MPE/iX. No access, opens the file or directory without any access checking. A process must be executing in system code to use this access type.
		9 Directory read access, opens a directory for directory read access. Directories can only be opened for no access or directory read access. Files cannot be opened for directory read access.
		FIFO files should be opened for Read Access Only (0) or Write Access Only (1). Other access types can cause unexpected results to occur.

**HPFOPEN**

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
12	I32	<p>Dynamic locking:</p> <p>Passes a value enabling/disabling file locking for the file. When specified, the <b>FLOCK</b> and <b>FUNLOCK</b> intrinsics can be used to dynamically permit/restrict concurrent access to a disk file by other processes at specified times.</p> <p>The following values are valid:</p> <p>0    Disallow dynamic locking/unlocking</p> <p>1    Allow dynamic locking/unlocking</p> <p>Default: 0</p> <p>The process can continue this temporary locking/unlocking until it closes the file. If several accessors are sharing the file, they must all specify, or not specify, this option. For example, if a file is opened with the dynamic locking option enabled, and a subsequent accessor tries to open the file with dynamic locking disabled, that subsequent attempt to open fails.</p> <p>Dynamic locking/unlocking is possible through the equivalent of a global resource identification number (RIN) assigned to the file and temporarily acquired by <b>HPFOPEN</b>.</p> <p>Accessors that have opened a file with the dynamic locking option enabled must access the file through the <b>FLOCK</b> and <b>FUNLOCK</b> intrinsics to gain exclusive access to the file. Since the use of these intrinsics is discretionary, however, all accessors must agree to use <b>FLOCK</b> and <b>FUNLOCK</b> when writing to a file to guarantee exclusive access. File locking is advised, but is not mandated by MPE/iX.</p> <p>Lock access must be at the account, group, and file levels for <b>HPFOPEN</b> to grant this option. (Lock access is available if lock, execute, append, or write access is set at these levels.) This option is ignored for files not residing on disk.</p> <p>This <i>itemnum</i> may only be specified with the disallow dynamic locking value (0) when used with directories.</p> <p>(ASC) Not used for asynchronous devices.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
13	I32	<p>Exclusive:</p> <p>Passes a value indicating continuous exclusive access to the file, from open to close. Use this option when performing a critical operation (for example, updating the file).</p> <p>The following values are valid:</p> <p>0 If <i>itemnum</i>=11 specifies read only access, read-share access takes effect. Otherwise, exclusive access takes effect. Regardless of which access option was selected, <b>FFILEINFO</b> reports zero. A zero (default) value for the <i>itemnum</i> specifies that if the access type is read, directory read, or no access (<i>itemnum</i> 11 equal to 0, 8, or 9) then shared access takes effect; otherwise exclusive access takes effect.</p> <p>1 Exclusive access. After the file is opened, any additional <b>HPFOPEN/FOPEN</b> requests for this file, whether issued by this process or another process, are prohibited until this process issues the <b>FCLOSE</b> request or terminates. If any process is already accessing this file when an <b>HPFOPEN/FOPEN</b> call is issued with exclusive access specified, an error status is returned to the process. If another <b>HPFOPEN/FOPEN</b> call is issued for this file while exclusive access is in effect, an error code is returned to the process that issued that <b>HPFOPEN/FOPEN</b> call. Request exclusive access only if the lock access mode is allowed by the security provisions for the file. For message files, specifying this value means that there can be only one reader and one writer.</p> <p>Exclusive access cannot be used with directories.</p> <p>2 Read-Share access (semi-exclusive access). After the file is opened, concurrent write access to this file through another <b>HPFOPEN/FOPEN</b> request is prohibited, whether issued by this process or another process, until this process issues the <b>FCLOSE</b> request or terminates. A subsequent request for the read/write or update <i>itemnum</i>=11 obtains read access. However, other types of read access are allowed. If a process already has write access to the file when this <b>HPFOPEN</b> call is issued, an error code is returned to the calling process. If another <b>HPFOPEN/FOPEN</b> call that violates the read-only restriction is issued while read-share access is in effect, that call fails and an error code is returned to the calling process. You can request read-share access only if you are allowed the lock access mode by the security provisions for the file. For message files, specifying this value means that there can be multiple readers, but only one writer.</p> <p>3 Share access. After the file is opened, this permits concurrent access to this file by any process, in any access mode, subject to other basic security provisions in effect. For message files, specifying this value means that there can be multiple writers and one reader.</p> <p>Default: 0</p> <p>(ASC) This option is ignored for devices.</p> <p>FIFO files should be opened for Share Access (3). Other exclusive accesses to file will cause unexpected results to occur.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
14	I32	<p>Multiaccess:</p> <p>Passes a value indicating how the file's record pointer is to be shared. This option is useful for sharing standard input devices where there is some natural sequence of access to the file. This option permits processes located in different jobs or sessions to open the same file and share that file's record pointer.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 No multiple process access allowed. A unique record pointer is created for this access to the file. For message files, the file system sets the multiaccess option to 2 when a zero is specified for this option.</li> <li>1 Intrajob multiprocess access allowed. A record pointer is shared with all other opened files of the same name in the same job/session who opened the file with <i>itemnum</i>=14 is set to either 1 or 2.</li> <li>2 Interjob multiprocess access allowed. A record pointer is shared with all other opened files of the same name on the system. This is the same as specifying the <b>GMULTI</b> option in a <b>FILE</b> command.</li> </ul> <p>Default: 0</p> <p>Native byte stream access (see item 77) is opened regardless of the value of this <i>itemnum</i>. This <i>itemnum</i> is also ignored for directories since it is not applicable.</p> <p>(ASC) Not used for asynchronous devices.</p>
15	I32	<p>Multirecord:</p> <p>Passes a value indicating that individual read or write requests are not confined to record boundaries.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 Nonmultirecord mode (<b>NOMULTI</b>)</li> <li>1 Multirecord mode (<b>MULTI</b>)</li> </ul> <p>Default: 0</p> <p>If the number of half words or bytes to be transferred (specified in the <i>length</i> parameter of the read or write request) exceeds the size of the physical record (that is, a block) that is referenced, the remaining half words or bytes are taken from subsequent successive records until the number specified by <i>length</i> has been transferred. For message (MSG) files not accessed with <i>itemnum</i>=17 enabled, the file system sets this option to zero. This option is available only if <i>itemnum</i>=46 is set to 1.</p> <p>(ASC) This option is not used for printers.</p>



Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
16	I32	<p>Nowait I/O:</p> <p>Allows the accessor to initiate an I/O request and to have control returned before the completion of the I/O. This option implies the inhibit buffering option; if <b>NOBUF</b> is not specified, the file system does it. Multirecord access is not available. This option is not available if the file is located on a remote computer. When opening nonmessage files, the process must be running in PM (execution level 2) to specify this option. Set <i>itemnum</i>=29 to 3 if the file is to be accessed while in user mode (execution level 3).</p> <p>The following values are valid:</p> <p>0    Nowait I/O not in effect</p> <p>1    Nowait I/O in effect</p> <p>Default: 0</p> <p>Directories may not be opened using Nowait I/O (1).</p>
17	I32	<p>Copy mode:</p> <p>Passes a value that determines if any file should be treated as a standard sequential file so it can be copied by logical record or physical block to another file.</p> <p>Byte stream files and directories are accessed using normal access (0) regardless of the value specified for this <i>itemnum</i>.</p> <p>The following values are valid:</p> <p>0    The file is accessed as its own file type (for example, a message file is treated as a message file).</p> <p>1    The file is to be treated as a standard (STD) file with variable-length records. For message files, this allows nondestructive reading of an old message file at either the logical record or physical block record level. Only block-level access is permitted if the file is opened with write access. This prevents incorrectly formatted data from being written to the message file while it is unprotected. To access a message file in copy mode, a process must have exclusive access to the file.</p> <p>(KSAM) Not allowed for KSAM XL files.</p> <p>Default: 0</p>

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
18	@32	<p>Short-mapped:</p> <p>Returns a short pointer to the beginning of the data area of the file. This option maps the file into short pointer space. A short-mapped file can be 4 megabytes in length. The calling process can have up to 6 megabytes of short-mapped files open at a time. Use the pointer as a large array of any type to efficiently access the file.</p> <p>A file previously opened normally (not mapped) or with the long-mapped option is not accessible with the short-mapped option. If this option is specified with the file already opened into long pointer space, an error results.</p> <p>A loaded program file or a loaded library file is not accessible with the short-mapped option. A file cannot be loaded that is currently opened with the short-mapped option.</p> <p>Sharing of short pointer files is provided through normal file system sharing mechanisms, for example, use of the exclusive option. With the short-mapped file, all file system intrinsics, applicable to the file, can be used. <b>FREAD</b> and <b>FWRITE</b> calls can be mixed with the short-mapped access.</p> <p>Standard (STD) type disk files of fixed or undefined record length can be accessed short-mapped with the access type option set to any value. Standard type disk files of variable record length can be accessed short-mapped only if the access type option is set to read-only access. KSAM files can be accessed short-mapped only if the access type option is set to read-only access and the copy mode option is set to 1.</p> <p>Directories may not be opened using short-mapped access.</p> <p>Default: No short pointer returned</p> <p>(ASC) Not used for asynchronous devices.</p>
19	I32	<p>Record size:</p> <p>Passes the size, in bytes, of the logical records in the file. Valid range is dependent upon both storage format (ASCII or binary) and record format. For fixed-length and undefined-length ASCII files, a record size can be specified in the range 1..32,767. For variable-length ASCII files, and for fixed-length, variable-length, and undefined-length binary files, a record size can be specified in the range 1..32,766.</p> <p><b>HPFOPEN</b> rounds up odd values to the next highest even number (equivalent to the nearest half word boundary) if the file is ASCII with variable-length record format, binary with fixed-length, variable-length, or undefined-length record format.</p> <p>For example, if a record size of 105 is specified for a fixed-length binary file, <b>HPFOPEN</b> sets the record size to 106; if a record size of 233 is specified for a fixed-length ASCII file, the record size remains the same as it was when specified.</p> <p>The value specified for this <i>itemnum</i> is ignored when a byte stream file or hierarchical directory is created. Byte stream files are created with a logical record size of one byte (1). Hierarchical directories are created with a logical record size of 32 bytes (32).</p> <p>Default: 256</p>
4-344	<b>Command Definitions</b>	<p>(ASC) For terminal and printer files, no rounding up occurs if a record size consisting of an odd number of bytes is specified. The record size can be different from the port configuration. The default is the configured record size (normally 40 words for terminals, 66 words for printers).</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
20	CA	<p>Device name:</p> <p>Passes the logical device number, in ASCII form, of a specific device. The file is assumed to be permanent. If the device name option is specified, the nonshareable device should be ready prior to the <b>HPFOPEN</b> call (otherwise, an error results).</p> <p>Only one of the following options can be in effect when a file is opened:</p> <p><i>itemnum=20</i>  <i>itemnum=22</i>  <i>itemnum=23</i>  <i>itemnum=42</i></p> <p>Default: Disk file located on the volume class <b>DISC</b> associated with the group in which file resides.</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example:</p> <p><i>%devname%</i> (% is the delimiter, <i>devname</i> is the designator)  <i>fabcrxyzf</i> (<i>f</i> is the delimiter, <i>abcrxyz</i> is the designator)</p>
21	@64	<p>Long-mapped:</p> <p>Returns a long-pointer to the beginning of data of the file. This option maps the file into long pointer space. A long-mapped file can be up to 2-gigabytes in size. There is no limit to the number of long-mapped files a process can have open at one time. The pointer can be used as a large array of any type to access the file.</p> <p>A loaded program file or a loaded library file is not accessible, and a file cannot be loaded with this option.</p> <p>Sharing long-pointer files is provided through normal file system file sharing mechanisms. All file system intrinsics applicable to the file can be used. <b>FREAD</b> and <b>FWRITE</b> calls can be mixed with this option.</p> <p>Standard (STD) disk files of fixed or undefined record length can be accessed with this option. Standard disk files of variable record length can be accessed only if <i>itemnum=11</i> (read-only access). Standard disk files of variable record length and KSAM files can be accessed only if <i>itemnum=11</i> (read-only access) and <i>itemnum=17</i> (set to 1).</p> <p>Directories may not be opened using long-mapped access.</p> <p>Default: Not returned</p> <p>(ASC) Not used for asynchronous devices.</p>

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
22	CA	<p>Volume class:</p> <p>Passes a character array representing a volume class name where the file space is to be restricted. This option is applicable only at file creation.</p> <p>A volume class is a subset of volumes within a volume set. The volume class name must be a valid volume class name residing on the volume set bound to the volume (the volume set is an attribute of the group in which the file resides).</p> <p>Only one of the following options can be in effect when a file is opened with this option:</p> <pre> itemnum=20 itemnum=22 itemnum=23 itemnum=42 </pre> <p>Default: A disk file located on the volume class <b>DISC</b> associated with the group in which file resides.</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example:</p> <pre> %volclass% (% is the delimiter, volclass is the designator) fabcxyzf (f is the delimiter, abcxyz is the designator) </pre> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories.</p> <p>(ASC) Not used for asynchronous devices.</p>
23	CA	<p>Volume name:</p> <p>Passes a character array representing a volume name that restricts the file specified to a specific volume. The volume must reside within the volume set of the group where the file resides. This option is applicable only at file creation.</p> <p>Only one of the following options can be in effect when a file is opened with this option:</p> <pre> itemnum=20 itemnum=22 itemnum=23 itemnum=42 </pre> <p>Default: A disk file located on the volume class <b>DISC</b> associated with the group in which the file resides.</p> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories.</p> <p>(ASC) Not used for asynchronous devices.</p>
24	I32	<p>Density:</p> <p>Passes the tape density required when writing to a tape file. This option is applicable only when writing to a tape on a drive that supports more than one density. When reading from a tape, the density of the tape overrides this option.</p> <p>Default: The highest density available on the device opened.</p> <p>(ASC) Not used for asynchronous devices.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
25	CA	<p>Printer environment:</p> <p>Passes the name of a file that contains a printer environment. This option is valid only for specified printer devices.</p> <p>If opening an Hewlett-Packard 268x page printer file, specify an optional printing environment for the job. The printing environment is defined as all of the characteristics of the printed page that are not part of the data itself, including the page size, the margin width, the character set, the orientation (horizontal or vertical), and the name of forms to use. If an environment file is not specified, the file system selects a default printer environment.</p> <p>Any environment selected remains active until replaced by a new environment, or until a call to <b>FCLOSE</b> (close the printer).</p> <p>Default: No printer environment file specified</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example,</p> <p><i>%envname%</i> (% is the delimiter, <i>envname</i> is the designator)  <i>fabcrxyzf</i> (f is the delimiter, <i>abcrxyz</i> is the designator)</p>
26	CA	<p>Remote environment:</p> <p>Passes the node name of the remote computer where the file is located. This option is used when referencing a file located on a remote computer.</p> <p>Default: No node name passed (local file access)</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example,</p> <p><i>%envname%</i> (% is the delimiter, <i>envname</i> is the designator)  <i>fabcrxyzf</i> (f is the delimiter, <i>abcrxyz</i> is the designator)</p> <p>A remote environment cannot be specified when creating or opening files in the HFS name space or in byte stream files.</p>
27	I32	<p>Output priority:</p> <p>Passes the output priority to be attached to the file for spooled output. This option is applicable only to spooled devices. The output priority must be a number between 1 (lowest priority) and 13 (highest priority), inclusive. If the value specified is less than the current outfence set by the system operator, file printing is deferred until the operator raises the output priority of the file or lowers the outfence. This option can be specified for a file already opened (for example, <b>\$STDLIST</b>), where the highest value supplied before the last <b>FCLOSE</b> takes effect. This option is ignored for nonspooled devices.</p> <p>Default: 8</p>

**HPFOPEN**

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
28	CA	<p>Spooled message:</p> <p>Passes a spooler message associated with a spoolfile. For example, a message is passed that can be used for telling the system operator what type of paper to use in the line printer. This message must be displayed to the system operator and verified before the file can be printed on a line printer. The number of characters allowed &lt;=48; any quantity &gt; 48 characters is truncated.</p> <p>Default: No spooled message specified</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example:</p> <p><i>%message%</i> (% is the delimiter, <i>message</i> is the designator)  <i>fabcryzf</i> (<i>f</i> is the delimiter, <i>abcryz</i> is the designator)</p>
29	I32	<p>Privileged access:</p> <p>Passes a value that temporarily restricts access to the file number returned from <b>HPFOPEN</b> to a calling process whose execution level is equal to or less than the value specified in this option. This restriction lasts until the file associated with the restricted file number is closed. Do not specify a value less than the execution level of the calling process.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 Privilege level zero (most privileged level)</li> <li>1 Privilege level one</li> <li>2 Privilege level two</li> <li>3 Privilege level three (least privileged level)</li> </ul> <p>Default: The execution level of the calling process</p>
30	I32	<p>Labeled tape type:</p> <p>Passes a value that indicates tape label type information. This option is valid only for labeled tapes. The following values are valid:</p> <ul style="list-style-type: none"> <li>0 ANSI standard labels</li> <li>1 IBM standard labels</li> </ul> <p>Default: 0</p> <p>(ASC) Not used for asynchronous devices.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
31	CA	<p>Labeled tape expiration:</p> <p>Passes the date of the expiration of the file or the date after which the information in the file is no longer useful, in the format <b>MM/DD/YY</b>. The file can be overwritten after this date. If the default is specified, the file can be overwritten immediately. In a volume set, file expiration dates must always be equal to or earlier than the date on the previous file.</p> <p>Default: No expiration date specified</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example,</p> <p><i>%expdate%</i> (% is the delimiter, <i>expdate</i> is the designator)  <i>fabcxyzf</i> (<i>f</i> is the delimiter, <i>abcxyz</i> is the designator)</p> <p>(ASC) Not used for asynchronous devices.</p>
32	CA	<p>Labeled tape sequence:</p> <p>Passes one of the following character arrays indicating the position of the file in relation to other files on the tape:</p> <p>0 Causes a search of all volumes until the file is found.</p> <p>1.. Specifies the position of the file relative to the current file on the tape.  9999</p> <p><b>ADDF</b> Causes the tape to be positioned so as to add a new file at the end of the volume or last volume in a multivolume set.</p> <p><b>NEXT</b> Positions the tape at the next file on the tape. If this is not the first <b>HPFOPEN/FOPEN</b> for the file and a rewind occurred on the last close, then the position remains at the beginning of the previous file.</p> <p>Default: No array passed (no sequence indicated)</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example,</p> <p><i>%position%</i> (% is the delimiter, <i>position</i> is the designator)  <i>fabcxyzf</i> (<i>f</i> is the delimiter, <i>abcxyz</i> is the designator)</p> <p>(ASC) Not used for asynchronous devices.</p>
33	I32	<p>User labels:</p> <p>Passes the number, in the range 0..254, of user-label records to be created for the file. Applicable for new disk files only.</p> <p>Default: 0</p> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories.</p> <p>(ASC) Not used for asynchronous devices.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
34	I32	<p>Spooler copies:</p> <p>Passes a value in the range 1..127 indicating the number of copies of the entire file to be produced by the spooling facility. This option is applicable to spooled devices only. This option can be specified for a file already opened (for example, \$STDLIST), where the highest value supplied before the last FCLOSE takes effect. The copies do not appear continuously if the system operator intervenes or if a file of higher output priority becomes READY before the last copy is complete. This option is ignored for nonspooled output devices.</p> <p>Default: 1</p>
35	I32	<p>File size:</p> <p>Passes the maximum file capacity:</p> <ul style="list-style-type: none"> <li>■ For variable-length records, the capacity is expressed in blocks (<math>blockitem\# = recordsize * blockfactor</math>).</li> <li>■ For fixed-length and undefined-length records, the capacity is expressed in logical records.</li> <li>■ The maximum file size for standard and KSAM XL files is 4,294,901,759 bytes (4 gigabytes less 64K bytes).</li> <li>■ The maximum file size for a CM KSAM data file is 2 gigabytes and for a CM KSAM key file, 2 gigabytes.</li> <li>■ The maximum file size of 500 megabytes, for RIO and circular is dependent upon both the record size and the number of extents defined for the file: <ul style="list-style-type: none"> <li>□ For circular and RIO files, <math>recsize=256</math> bytes and <math>numextent=32</math>.</li> </ul> </li> </ul> <p>This option is applicable only at file creation.</p> <p>Default: 2 gigabytes</p> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories. (ASC) Not used for asynchronous devices.</p>
36	I32	<p>Initial allocation:</p> <p>Passes a positive integer value indicating the number of records to be allocated to the file initially. This option is applicable only at file creation.</p> <p>Default: 0</p> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories. (ASC) Not used for asynchronous devices.</p>



Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
37	I32	<p>File code:</p> <p>Passes a value that can be used as a file code to identify the type of file. This code is recorded in the file label and is accessible through the <b>FFILEINFO</b> intrinsic. This option is applicable only at file creation (except when opening an old file that has a negative file code).</p> <p>If the program is running in user mode, specify a file code in the range 0..32,767 to indicate the file type being created. Programs running in user mode can access files with positive file codes only.</p> <p>If the program is running in privileged mode, specify a file code in the range -32,768..32,767. Programs running in privileged mode can access files with a file code in the range -32,768..32,767. If an old file is opened that has a negative file code in its file label, the file code specified must match the file code in the file label (otherwise, an error results).</p> <p>File codes cannot be specified for hierarchical directories. Negative file codes may be used only for files in MPE groups. A <i>status.info</i> of <b>-315</b> is returned when negative file codes are specified for files outside MPE groups.</p> <p>Default: 0</p> <p>(ASC) Not used for asynchronous devices.</p>
38	I32	<p>File privilege:</p> <p>Passes a value that determines a permanent privilege level to be associated with a newly created file. This option permanently restricts file access to a process whose execution level is less than or equal to the specified value. A value cannot be specified for less than the execution level of the calling process. This option is applicable only at file creation.</p> <p>The following values are valid:</p> <ul style="list-style-type: none"> <li>0 Privilege level zero (most privileged level)</li> <li>1 Privilege level one</li> <li>2 Privilege level two</li> <li>3 Privilege level three (least privileged level)</li> </ul> <p>Default: 3</p>

**HPFOPEN**

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
39	I32	<p>Access type:</p> <p>Passes a value indicating how to use the file, either sequentially or randomly. The file system uses this information to determine the most efficient prefetching algorithm to improve the performance of the file access.</p> <p>The following values are valid:</p> <p>0     Access the file sequentially</p> <p>1     Access the file randomly</p> <p>Default: 0</p> <p>(ASC) Not used for asynchronous devices.</p>
40	I32	<p>Block factor:</p> <p>Passes the number of logical records to be contained in one physical record (block). This value is used to calculate the physical record size (block size) for disk and magnetic tape files. Valid ranges are 1..32,767. This option is applicable only at file creation.</p> <p>For fixed-length records, this option specifies the actual number of records in a block. For variable-length records, this option is interpreted as a multiplier used to compute the block size (<i>record size option * block factor option</i>). For undefined-length records, this option is always one logical record per block.</p> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories.</p> <p>Default: 1 for files opened <b>NOBUF</b>; for files opened <b>BUF</b>, it is calculated by dividing the specified records into the block size configured for the device.</p> <p>(ASC) Not used for asynchronous devices.</p>
41	I32	<p>Name syntax:</p> <p>Specifies which of three name semantics will be used to interpret the filename passed to <b>HPFOPEN</b>:</p> <p>0                   MPE-escaped semantics</p> <p>1                   MPE-only semantics</p> <p>2                   POSIX semantics</p> <p>MPE-escaped name semantics is the default value for this <i>itemnum</i>. The selected name semantics do not apply to file equations specified through <i>itemnum</i> 52 or system-defined file references specified through <i>itemnum</i> 5.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
42	CA	<p>Device class:</p> <p>Passes a device class where the file resides. The file system uses the device class name to select a nonshareable device from a configured list of available devices. The name can have a length of up to eight alphanumeric characters, beginning with a letter (for example, <b>TAPE</b>). If a device class is specified, the file is allocated to any available device in that class.</p> <p>Only one of the following options can be in effect when a file is opened:</p> <pre> itemnum=20 itemnum=22 itemnum=23 itemnum=42 </pre> <p>Default: A disk file located on the volume class <b>DISC</b> associated with the group in which file resides.</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example:</p> <pre> %devclass% (% is the delimiter, devclass is the designator) fabcrxyzf (f is the delimiter, abcrxyz is the designator) </pre> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories. Hierarchical directories created on the system volume set is allocated on any volume within the set. Hierarchical directories created on non-system volume sets is allocated on the master volume.</p>
43	record	<p>UFID:</p> <p>Passes a unique file identifier (UFID) to provide a fast opening of an old disk file. A UFID is a record structure, 20 bytes in length, that uniquely identifies a disk file. Using this option avoids a directory search. Obtain the UFID of an opened file by calling <b>FFILEINFO</b>. The UFID can then be passed to <b>HPFOPEN</b>. The file represented by the UFID must be accessible to the process calling <b>HPFOPEN</b>. (All file system security checks are made.) New files cannot be opened with this option. If the file to be opened by the UFID contains a lockword, use <i>itemnum=2</i> to specify the file name with the lockword.</p> <p>Only files in the MPE name space may be opened by UFID. An attempt to open a file outside the MPE name space by UFID results in a <i>status.info</i> of -321 being returned. Only system code may open a file by UFID in the POSIX name space.</p> <p>Default: No UFID passed (a directory search is performed)</p> <p>(ASC) Not used for asynchronous devices.</p>

**HPFOPEN**

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
44	I32	<p>Numbuffers:</p> <p>Passes the number of buffers to allocate to the file. Ignored for standard disk files. This option is useful only for slow devices (such as tapes) used in a buffered mode. Not applicable for files representing interactive terminals; a system-managed buffering method is always used.</p> <p>The valid range for this option is dependent upon the file type:</p> <ul style="list-style-type: none"> <li>■ For standard and KSAM files, the valid range is 1..31.</li> <li>■ For circular and RIO files, the valid range is 1..16.</li> <li>■ For message files, the valid range is 2..16. (If a 1 is specified, the file system sets this option to 2 and no error is returned.)</li> </ul> <p>This option must not specify a number of buffers whose combined size exceeds the physical capacity of the file.</p> <p>This <i>itemnum</i> is ignored when creating hierarchical directories.</p> <p>Default: 2</p> <p>(ASC) Not used for asynchronous devices.</p>
45	CA	<p>Fill character:</p> <p>Passes two ASCII characters that determine what padding character to use at the end of blocks or unused pages, and the padding used by <i>itemnum</i>=53. Do not use delimiter characters for this option. The fill character must be a 2 byte array. The first character only is used as the padding character. The second character is reserved for future use. This option is applicable only at file creation.</p> <p>Default: Null characters for a binary file and ASCII blanks for an ASCII file.</p> <p>The default fill character for byte stream files is the null character. The fill character is null for hierarchical directories regardless of the value specified for this <i>itemnum</i>.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
46	I32	<p>Inhibit buffering:</p> <p>Passes a value enabling/disabling automatic operating system buffering. If <b>NOBUF</b> is specified, I/O is allowed to take place directly between the data area and the applicable hardware device.</p> <p>The following values are valid:</p> <p>0 Allow normal buffering (<b>BUF</b>)</p> <p>1 Inhibit buffering (<b>NOBUF</b>)</p> <p>Default: 0</p> <p><b>NOBUF</b> access is oriented to physical block transfer rather than logical record transfer. If <b>NOBUF</b> and a variable-length record structure are specified in <i>itemnum=6</i> and the file does not have a variable-length record format, then the format is changed internally to an undefined-length record format. When performing an <b>FWRITE</b>, set up the variable structure.</p> <p>With <b>NOBUF</b> access, responsibility for blocking and deblocking of records in the file belongs to the program. To be consistent with files built using buffered I/O, records should begin on half word boundaries. When the information content of the record is less than the defined record length, you must pad the record with blanks if the file is ASCII, or with zeros if the file is binary.</p> <p>The record size and block size for files manipulated with <b>NOBUF</b> specified follow the same rules as those files that are created using buffering. The default blocking factor for a file created under <b>NOBUF</b> is one.</p> <p>If a file is opened <b>NOBUF</b> without multirecord mode specified in <i>itemnum=15</i>, then transfer a maximum of only one block of data per read or write.</p> <p>The end-of-file (EOF) marker, next record pointer, and record transfer count are maintained in terms of logical records for all files. The number of logical records affected by each transfer is determined by the size of the transfer.</p> <p>Transfers always begin on a block boundary. Those transfers that do not transfer whole blocks leave the next record pointer set to the first record in the next block. The EOF marker always points at the last record in the file.</p> <p>For files you have opened <b>NOBUF</b>, the <b>FREADDIR</b>, <b>FWRITEDIR</b>, and <b>FPOINT</b> intrinsics treat the <i>recnum</i> parameter as a block number.</p> <p>Indicate non-RIO access to an RIO file by specifying the file <b>NOBUF</b>. Use the physical block size from <b>FFILEINFO</b> to determine the maximum transfer length.</p> <p>For message files, the file system normally resets <i>itemnum=46</i> to zero. However, a message file can be opened with <b>NOBUF</b> if <i>itemnum=17</i> is set to 1; this determines access to the file record-by-record or by block.</p> <p>If writing to a message file, open it <b>NOBUF</b> if <i>itemnum=17</i> to access the file block-by-block.</p> <p>Native byte stream access opens <b>NOBUF</b> (1) regardless of the value specified for this <i>itemnum</i>. This <i>itemnum</i> is ignored for directories since a physical block transfer interface is not provided for directories.</p> <p>(ASC) Not used for asynchronous devices.</p>

## HPFOPEN

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
47	I32	<p>Numextents:</p> <p>Passes a value in the range 1..32 that determines the number of extents for the file. If a value of 1 is specified, the file is created as one contiguous extent of disk space. If a value greater than 1 is specified, a variable number of extents (with varying extent sizes) is allocated on a need basis. This option is applicable only at file creation.</p> <p>(ASC) Not used for asynchronous devices.</p> <p>Default: 1</p> <p>This <i>itemnum</i> may not be specified when creating hierarchical directories.</p>
48	I32	<p>Reverse VT:</p> <p>Passes a value indicating whether or not the given device name is to be allocated on a remote machine. Specify the remote environment in the same open request, using either the formal designator option or the remote environment option. Reverse VT behaves nearly the same as a terminal opened through remote file access, except that no session is required on the remote machine.</p> <p>The following values are valid:</p> <p>0    No reverse VT</p> <p>1    Reverse VT</p> <p>Default: 0</p>
49		Reserved for the operating system

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
50	I32	<p>Final disposition:</p> <p>Passes a value indicating the final disposition of the file at close time (significant only for files on disk and magnetic tape). A corresponding parameter in a <b>FILE</b> command can override this option, unless file equations are disallowed with <i>itemnum=9</i>.</p> <p>The following values are valid:</p> <p>0 No change. The disposition remains as it was before the file was opened. If the file is new, it is deleted by <b>FCLOSE</b>; otherwise, the file is assigned to the domain it belonged to previously. An unlabeled tape file is rewound. If the file resides on a labeled tape, the tape is rewound and unloaded.</p> <p>1 Permanent file. If the file is a disk file, it is saved in the system file domain. A new or temporary file on disk has an entry created for it in the system (permanent) file directory. If a file of the same name already exists in the directory, an error code is returned at close time and the file remains open. If the file is a permanent file on disk, this domain disposition has no effect. If the file is stored on magnetic tape, the tape is rewound and unloaded.</p> <p>2 Temporary job file (rewound). The file is retained in your temporary (job/session) file domain and can be requested by any process within your job/session. If the file is a disk file, the uniqueness of the file name is checked. If a file of the same name already exists in the temporary file domain, an error code is returned at close time and the file remains open. When a file resides on unlabeled magnetic tape, the tape is rewound. However, if the file resides on labeled magnetic tape, the tape is backspaced to the beginning of the presently opened file.</p> <p>3 Temporary job file (not rewind). This value has the same effect as specifying final disposition option <i>itemnum=2</i>, except that tape files are not rewind. In the case of unlabeled magnetic tape, if the <b>FCLOSE</b> is the last done on the device (with no other <b>FOPEN/HPFOPEN</b> calls outstanding), the tape is rewound and unloaded. If the file resides on a labeled magnetic tape, the tape is positioned to the beginning of the next file on the tape.</p> <p>4 Released file. The file is deleted from the system.</p> <p>5 Convert a permanent file to a temporary file. The file is removed from the permanent file directory and placed in the temporary file directory. (Privileged mode capability is required to use this option.)</p> <p>Default: 0</p> <p>For more information on file disposition at close time, refer to the description of the <b>FCLOSE</b> intrinsic.</p> <p>(ASC) Not used for asynchronous devices.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
51	String	<p>Pascal XL string:</p> <p>Passes a formal file designator that follows MPE/iX file naming conventions, using the Pascal XL STRING type format. This option is identical to <i>itemnum=2</i> except for the type of item. No delimiters are needed.</p> <p>Default: No string passed</p> <p>When you use <b>HPFOPEN</b> to open a file, you may use either <i>itemnum=2</i> or you may use <i>itemnum=51</i>; you may not use both.</p> <p>As the default, the formal file designator is interpreted according to MPE-escaped semantics. To choose another syntax, use <i>itemnum 41</i>.</p> <p>The file referred to by <i>formaldesig</i> can be either an MPE file (i.e., one that uses MPE syntax) or it can follow HFS syntax. If <i>formaldesig</i> follows MPE syntax, the file name can include password, group, and account specifications. The file name can backreference a file equation and optionally be preceded by an asterisk. If <i>formaldesig</i> follows HFS syntax, the file name must start with either a dot (.) or a slash (/).</p> <p>The file referred to by <i>formaldesig</i> may reside either in an MPE group or in an HFS directory. For files located in HFS directories, traverse directory entries (TD) access is required to all directories specified in <i>formaldesig</i>. If there is no TD access, <b>HPFOPEN</b> fails and a value of -180 is returned in the <i>status.info</i> parameter. If you are using <b>HPFOPEN</b> to create a file or hierarchical directory and you lack of create directory entry (CD) access, <i>status.info</i> will return a value of -179.</p>
52	CA	<p>File equation string:</p> <p>Passes a character string that matches the file equation specification syntax exactly. (Refer to the <b>FILE</b> command in the <i>MPE/iX Commands Reference Manual Volumes 1 and 2</i> (32650-90003 and 32650-90364).) This option allows the specification of options available in the <b>FILE</b> command.</p> <p>The <i>formaldesig</i> parameter and <i>flereference</i> parameter can contain embedded command interpreter variables and expressions. However, there cannot be more than 8-characters in each of these components (<i>filename</i>, <i>lockword</i>, <i>groupname</i>, <i>accountname</i>) including the command interpreter variable and expression characters.</p> <p>Default: No string passed</p> <p>A character placed in the first element designates the delimiter used by <b>HPFOPEN</b> to search for the end of the character array. The delimiter can appear again only following the last valid character of the character array, for example:</p> <p style="padding-left: 40px;"><i>%fileequation%</i> (% is the delimiter, <i>fileequation</i> is the designator)  <i>fabcxyzf</i> (f is the delimiter, <i>abcxyz</i> is the designator)</p> <p>The <i>formaldesignator</i> in the file equation string must belong to the MPE name space. The <i>flereference</i> in the file equation string is interpreted using MPE-escaped semantics. Both the <i>formaldesignator</i> and the <i>flereference</i> in the file equation string may also contain embedded command interpreter variables or expressions.</p>



Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
53	I32	<p>ASCII/binary:</p> <p>Passes a value indicating whether ASCII or binary code is to be used for a new file when it is written to a device that supports both codes. For disk files, this affects padding that can occur when issuing a direct-write intrinsic call (<b>FWRITEDIR</b>) to a record that lies beyond the current logical end-of-file indicator. By default, magnetic tape and files are treated as ASCII files. This option is applicable only at file creation.</p> <p>The following values are valid:</p> <p>0    Binary file</p> <p>1    ASCII file</p> <p>Default: 0</p> <p>(ASC) Not used for asynchronous devices.</p>
54	REC	<p>KSAM parm:</p> <p>Passes a record that defines the keys for a new KSAM file.</p> <p>(KSAM XL) For KSAM XL files, refer to the <i>parm</i> parameter discussion in the <i>Using KSAM XL</i> (32650-90168).</p> <p>(KSAM/3000) The record must be at least 34 bytes in size. For details, refer to the <i>ksamparam</i> parameter discussion in the <i>KSAM/3000 Reference Manual</i> (30000-90079).</p> <p>Default: No record passed</p> <p>(ASC) Not used for asynchronous devices.</p>
55		Reserved for the operating system
56	I32	<p>Object class:</p> <p>Passes a user object class number, in the range 0..10, that is associated with the file.</p> <p>Default: Determined by the file code for system and subsystem files, and by the file type and record type for normal user files.</p>
57		Reserved for the operating system

**HPFOPEN**

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
58		Reserved for the operating system
59		Reserved for the operating system
60		Reserved for the operating system
61		Reserved for the operating system
64	BA	<p>Access Control Definition:</p> <p>Passes a byte array defining the access control definition (ACD) to be attached to a new file. The byte array has a length of 1 to 279 bytes. Unlike other <b>HPFOPEN</b> options that expect a delimiter at both the beginning and the end of the byte array, this option only expects a trailing carriage return character as a delimiter, for example,</p> <pre data-bbox="492 720 1036 747">(X:@.@;R,W:MGR.SYS;RACD:JOHN.SMITH)&lt;cr&gt;</pre> <p>Where, the &lt;cr&gt; is the carriage return character (13, 0x0D).</p> <p>The ACD assigned to a newly created file or directory may be different from the ACD specified as the value of this <i>itemnum</i>. If a process' file mode creation mask (<b>cmask</b>) is initialized, it modifies the ACD.</p>
74		<p>Header/Trailer:</p> <p>Enable/disable standard headers/trailers:</p> <ul style="list-style-type: none"> <li>0 Enable header and trailer (default)</li> <li>1 Disable header</li> <li>2 Disable trailer</li> <li>3 Disable header and trailer</li> </ul> <p>Provides a standard way for the spooler to control the printing banners.</p> <p>Valid only for the initial <b>HPFOPEN</b> call to the printer and prevents the header from being printed. On the last <b>FCLOSE</b>, the trailer is not printed.</p>

Table 4-20. HPFOPEN Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
77	I32	<p>Data format</p> <p>Allows the caller to select a different format to view the data in the file. The current valid values for this item are:</p> <p>0                    Use the standard record based view of accessing the file. This is the default value for all opens. For conventional files, the file is record based. For directories, the standard non-privileged directory information is returned when the <b>HPDIRREAD</b> intrinsic is called. When this value is specified with a byte stream file, access to the file is emulated to appear like a buffered variable record file. This is the default.</p> <p>1                    When this value is specified, calls to <b>HPDIRREAD</b> returns privileged directory information including the UFID and link ID of the entry. Note that this value is only applicable to directory files. This value is ignored for all other file types. In order to specify this value, the caller must be executing in system code.</p> <p>2                    When this value is specified, the system attempts to let the caller access the file as a native byte stream file. Byte stream emulation is supported for ordinary fixed and variable length record files as well as for files with the byte stream record type. If this value is requested against a file type that does not support byte stream access, an error is returned.</p> <p>Specifying any value other than the values described above will result in an error.</p>
79	I32	<p>POSIX Non-Block Mode</p> <p>Specifying this itemnum allows the caller to open a file with the POSIX Non-Block mode. This item is useful for a subset of files (including pipes and FIFO's) and is ignored for all other files. The behavior of the <b>HPFOPEN</b> call with this option is dependent on the type of file being opened.</p> <p>The current valid values for this item are:</p> <p>0                    This value indicates that Non-Block mode is off. This is the default value.</p> <p>1                    This value indicates that Non-Block mode is on.</p> <p>Specifying any value other than those described above will result in error.</p>

## HPFOPEN

**Table 4-20. HPFOPEN Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
80	I32	Reserved for the operating system.
81	I32	<p>Symbolic link option:</p> <p>This <i>itemnum</i> allows the caller to specify different options when traversing through or opening a symbolic link. The valid values for this <i>itemnum</i> are described below:</p> <p>0                      Follow symbolic links. This is the default value for this option. When a symbolic link is encountered it is traversed according to the path specified in the symbolic link.</p> <p>1                      Does not follow symbolic links. If the final component of a path is a symbolic link, then no traversal is done and the symbolic link is opened. Symbolic links that occur prior to the last component is traversed.</p> <p>Specifying any value other than those above will result in error.</p>

## Operation Notes

Enables creation of a new file on a shareable device and defines the physical characteristics of that file prior to access. Enables access to existing files. Returns a file number to the calling process that uniquely identifies the file. Use the file number to reference the file in calls to other intrinsics.

## Related Information

Intrinsics      FOPEN

Commands      None

Manuals      *Accessing Files Programmer's Guide* (32650-90017), *Using KSAM XL* (32650-90168), and *KSAM/3000 Reference Manual* (30000-90079)

---

## HPFP\_CONVERT

NM and CM callable.

Converts data between binary floating-point formats.

### Syntax

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

### Parameters

*source*      **any supported real type by reference (required)**

Passes the floating-point number to be converted. Constants are not acceptable. You can specify a reference parameter of any supported real type in the intrinsic call statement. The format is identified by *sformat*.

*destination*      **any supported real type by reference (required)**

Returns the converted floating-point number. If *status.info* is less than zero, the number is not converted. Any supported real type can be returned to the intrinsic call statement. The format is identified by *dformat*.

*sformat*      **16-bit signed integer by value (required)**

The floating-point format of the binary number supplied by *source*. This is the format of the original number. Valid formats are:

Value	Meaning
1	Hewlett-Packard 3000 32-bit
2	Hewlett-Packard 3000 64-bit
3	IEEE 32-bit
4	IEEE 64-bit
5	IEEE 128-bit

***dformat*** 16-bit signed integer by value (required)

The destination floating-point format. This is the looking forward to some leisure and alternative nursing choices.format of the converted number. Valid formats are:

Value	Meaning
1	Hewlett-Packard 3000 32-bit
2	Hewlett-Packard 3000 64-bit
3	IEEE 32-bit
4	IEEE 64-bit
5	IEEE 128-bit

***status*** 32-bit signed integer by reference (optional)

Returns the status of the HPFPCONVERT intrinsic call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value. A positive value indicates that the conversion was made, but an exception occurred and an *exceptions* parameter was not supplied. The exception conditions are listed in the *exceptions* parameter description.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPFPCONVERT is 193.

## HPFPCONVERT

*exceptions*      **16-bit signed integer by reference (optional)**

Returns any exception conditions that occurred during the conversion. If *status.info* is less than zero, then *exceptions* is not modified because no conversion is done. If this parameter is not present, the value is returned in *status.info*.

Only the following values can be returned:

Value	Meaning
0	No exceptions
1	Inexact only (native mode <i>destination</i> only)
3	Underflow and inexact
5	Overflow and inexact
8	Invalid operation

If underflow or overflow occurs during conversion then inexact is also signaled.

*roundmode*      **16-bit unsigned integer by value (optional)**

The rounding mode (not applicable if *destination* is an Hewlett-Packard 3000 format).

Value	Meaning
0	Round to nearest, to even if tie (default)
1	Round to zero
2	Round to positive infinity
3	Round to negative infinity

### Operation Notes

The HPFPCONVERT intrinsic accepts a source binary floating-point number and converts it to the equivalent destination binary floating-point format. You must specify the format of the source and destination numbers. Optionally, you can specify the rounding mode.

The conversion is performed by regarding the source number as infinitely precise and with unbounded range, and then rounding it to fit the designated destination format. Rounding is performed according to the formal rules for the rounding mode specified. Rounding methods and exception signaling are determined solely from the destination format and are independent of the source format.

Conversion is performed as if all arithmetic traps are disabled. No trapping to user-supplied or system-supplied arithmetic trap routines is done.



**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Data Types Conversion Programmer's Guide</i> (32650-90015).

---

## HPGETPROCPLABEL

NM callable only.

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

### Syntax

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

### Parameters

*procname*     **character array (required)**

Passes the name of the procedure being searched for. The first character of *procname* designates the terminating character that HPGETPROCPLABEL uses to search for the end of the name. That delimiter can appear again only following the last valid character of the procedure name.

*plabel*        **32-bit unsigned integer by reference (required)**

Returns a procedure label (NM *plabel*) for the procedure that was found.

*status*        **32-bit signed integer by reference (optional)**

Returns the status of the HPGETPROCPLABEL intrinsic call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPGETPROCPLABEL is 104.

*firstfile***character array (optional)**

Passes the name of the program file or XL at which to begin searching. The *firstfile* parameter can be either in MPE syntax or HFS syntax. If the name of the program file or XL is in HFS syntax, you must use an absolute pathname. The first character that HPGETPROCPLABEL uses to search for the end of the file name. That delimiter can appear again only following the last valid character of the name.

If *firstfile* is located in the binding sequence of the calling process, then, beginning with *firstfile*, HPGETPROCPLABEL directs the NM loader to search through each file in the binding sequence for the first instance of a procedure named *procname*.

If either *procname* is not located in the XL or *procname* contains unresolved external references that must be satisfied in subsequent XLs, then each XL in the binding sequence after *firstfile* is searched.

If *firstfile* is not located in the binding sequence, then *firstfile* has a binding sequence independent of the program and is bound directly to the system libraries. Any unresolved external references found within *firstfile* must be resolved in the system libraries.

Default: System libraries

*casesensitive***Boolean (optional)**

Indicates in the following manner whether the *procname* parameter is case sensitive:

**FALSE** HPGETPROCPLABEL first tries finding *procname* as it was specified. If *procname* cannot be found, HPGETPROCPLABEL tries to find the procedure with the case of *procname* the opposite case of the first character of *procname* for example, if you specify *procname* as *scanforkey* and the procedure *scanforkey* is not found, the intrinsic searches for the procedure *SCANFORKEY*.

Default: FALSE

**TRUE** The *procname* parameter is case-sensitive. HPGETPROCPLABEL does not alter the name in any way.

**Operation Notes**

The HPGETPROCPLABEL intrinsic locates a procedure found in an NM executable library file (XL) and returns its procedure label (NM plabel). In addition, if the procedure is not yet loaded for the process, HPGETPROCPLABEL dynamically loads the procedure.

You can then use the NM plabel to call the specified procedure dynamically, provided the programming language contains features for making dynamic procedure calls.

**Note**

If you are going to use HPGETPROCPLABEL to call system intrinsics, you must first refer to the SYSINTR file to determine the correct parameter descriptions (for the parameters to be passed with the returned PLABEL).

## **HPGETPROCPLABEL**

A label returned by HPGETPROCPLABEL is valid only for the duration of the calling process.

### **Related Information**

Intrinsics        None

Commands        None

Manuals         *Resource Management Programmer's Guide* (32650-90024).

## HPLOADCMPROCEDURE

NM callable only.

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

### Syntax

U16	CA	U16V	I32
<i>plabel</i> :=HPLOADCMPROCEDURE( <i>procname</i> , <i>library</i> , <i>status</i> );			

### Functional Return

*plabel*            **16-bit unsigned integer (assigned functional return)**

Plabel of the target CM procedure.

### Parameters

*procname*        **character array (required)**

Passes an ASCII procedure name, left-justified and blank-padded. The name can have a maximum of 16 characters.

*library*          **16-bit unsigned integer by value (required)**

Passes indicator of the CM segmented library to be searched. The valid values are:

Value	Meaning
0	Search the system SL only. (Default)
1	Search the logon account SL, then the system SL.
2	Search the logon group SL, then logon account SL, and then system SL last.
3	Search the program file's account SL, then the system SL.
4	Search the program file's group SL, then the program file's account SL, and then the system SL last.

## HPLOADCMPROCEDURE

*status*            **32-bit signed integer by reference (optional)**

Returns the status of the HPCLOADCMPROCEDURE intrinsic call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPCLOADCMPROCEDURE is 105.

### Operation Notes

The HPCLOADCMPROCEDURE intrinsic helps you avoid switching by name overhead. Switching by name incurs the overhead of forming a hash probe out of the target procedure's name and finding the name in a hash table in order to obtain the procedure's plabel. You can eliminate this overhead by obtaining the target procedure's plabel through the HPCLOADCMPROCEDURE intrinsic. You can then supply the result returned by HPCLOADCMPROCEDURE as the value of the *procname* parameter in the call to the HPSWITCHTOCM intrinsic.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)

## HPLOADNMPROC

CM callable only.

Returns the plabel of an NM procedure.

### Syntax

U32	CA	I16V	CA
<i>plabel</i> := HPLoadNMProc( <i>procname</i> , <i>proclen</i> , <i>libname</i> ,			
I16V			
<i>liblen</i> )			

### Functional Return

*plabel*            **32-bit unsigned integer (assigned functional return)**

Label of the target NM procedure.

### Parameters

*procname*        **character array by reference (required)**

Passes the target procedure name. The target procedure must be contained in an executable library (XL). If the value of *procname* is invalid, blank, or does not contain the NM procedure, NL.PUB.SYS is searched.

*proclen*          **16-bit signed integer by value (required)**

Passes the byte length of the procedure name.

*libname*          **character array by reference (required)**

Passes the name of the NM library to be searched for the target. If the value of *libname* is invalid, blank, or does not contain the NM procedure, NL.PUB.SYS is searched.

*liblen*           **16-bit signed integer by value (required)**

Passes the byte length of the library name.

### Operational Notes

Do not use HPLoadNMProc to invoke a native mode (NM) system-supplied intrinsic. The result of the call may be unpredictable.

Used to obtain the NM plabel of the target procedure. Then the plabel is used by the HPSWTONMPLABEL intrinsic for its *proc* parameter.

The strings you supply as values of the *procname* and *libname* parameters must exactly match the names of the target NM routine and its NM library, respectively.

## HPLOADNMPROC

### Related Information

Intrinsics	HPSWITCHTOCMPROCEDURE
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)



---

## HPMERGEEND

NM callable only.

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

### Syntax

I32    I32A HPMERGEEND( <i>status,statistics</i> );
--

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPMERGEEND call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMERGEEND is 196.

*statistics*

**32-bit signed integer array (optional)**

Returns information on the Hewlett-Packard MERGE operation, as follows:

Value	Meaning
1	Number of input files
2	Number of merged records
3	Number of bytes used for the MERGE/XL work area
4	CPU time in milliseconds
5	Elapsed time in milliseconds

---

### Note

The *statistics* array should be the same array that was passed to the HPMERGEINIT intrinsic. It must be referenced by the same name.

---

### Related Information

Intrinsics	Hewlett-Packard SORT, Hewlett-Packard MERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## HPMERGEERRORMESS

NM callable only.

Accepts Hewlett-Packard MERGE intrinsic error code values and returns the error messages associated with them.

### Syntax

I32	CA	I32
HPMERGEERRORMESS( <i>status</i> , <i>message</i> , <i>length</i> );		

### Parameters

*status*

**32-bit signed integer by reference**

Passes the status of a previous Hewlett-Packard MERGE intrinsic call that failed. The *status* value passed in is used to locate the appropriate message catalog entry associated with the *status* value.

If no errors or warnings were encountered, *status* returns 32-bits of zero. If errors or warnings were encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for **HPMERGEERRORMESS** is 196.

---

### Note

HPMERGEERRORMESS returns information on the success of its execution in the *status* parameter. It is good programming practice to specify this parameter and check its value after the intrinsic call. If an error or warning condition is encountered, and you did not specify the *status* parameter, **HPMERGEERRORMESS** causes the calling process to abort.

---

*message*

**character array (optional)**

Returns the text of the error message. This parameter may be up to 80 characters long.

*length*

**32-bit signed integer by reference (optional)**

Returns the length of the *message* parameter.

**Related Information**

Intrinsics	Hewlett-Packard SORT, Hewlett-Packard MERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## HPMERGEINIT

NM callable only.

Initializes the MERGE/XL subsystem.

### Syntax

```
HPMERGEINIT( I32  I32A  PROC  I32A
              PROC  32V  I32V
              postprocessor, keysonly, numkeys,
              I32A CA  PROC  PROC
              keys, altseq, keycompare, errorproc,
              I32A  I32V  I32A
              statistics, memsize, charseq);
```

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPMERGEINIT call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMERGEINIT is 196.

*inputfiles*

**32-bit signed integer array (optional)**

Passes the file identification numbers of the input files to be merged. The last element of this array should be set to zero. If the files are opened with either the NOBUF or MR (multirecord) access option, MERGE/XL program performs the buffering and blocking/deblocking.

*preprocessor*

**procedure (reserved)**

Do not specify this parameter, but maintain parameter position.

- outputfiles*     **32-bit signed integer array (optional)**
- Passes the file identification number of the output file in the first array element. If the file is opened with either the **NOBUF** or **MR** (multirecord) access option (*aoption*), **MERGE/XL** performs the buffering and blocking/deblocking.
- If you do not specify this parameter, you must call the **HPMERGEOUTPUT** intrinsic to retrieve merged records.
- postprocessor*     **procedure (reserved)**
- Do not specify this parameter, but maintain parameter position.
- keysonly*     **32-bit signed integer by value (optional)**
- Flag that determines if only keys are sent as output. If set to 0, the entire record is sent as output. If set to any value other than 0, the key fields are concatenated together with the major key on the left and the remaining keys following. Default: 0
- numkeys*     **32-bit signed integer (optional)**
- Number of keys used during the comparison of records. This parameter can be either equal to or greater than one. If you specify the *numkeys* parameter, you must also specify the *keys* parameter. Together, *numkeys* and *keys* describe the way records are merged.
- keys*     **32-bit signed integer array (optional)**
- Passes information about the keys used during the comparison of records. If you specify the *keys* parameter, you must specify the *numkeys* parameter. Together, *keys* and *numkeys* describe the way records are merged.

## HPMERGEINIT

For each key, there are four entries in the array. The first element gives the position of the first byte of the key within the input record. The second element gives the number of bytes in the key. The third element indicates the type of data:

Value	Meaning
0	Byte type (ASCII or EBCDIC)
1	Twos complement (including 16-bit and 32-bit signed integer)
2	Hewlett-Packard 3000 floating-point format (including real and long)
3	IEEE standard floating-point format (32-bit, 64-bit, and 128-bit values)
4	Packed decimal with odd number of digits
5	Packed decimal with even number of digits
6	Display trailing sign
7	Display leading sign
8	Display trailing sign separate
9	Display leading sign separate
10	Character
11	Reserved for the operating system
12	Short floating-point decimal
13	Floating-point decimal

The fourth element gives the ordering sequence of the records:

Value	Meaning
0	ascending
1	descending

---

**Note** The integrity of the *keys* array must be maintained throughout the MERGE/XL operation. Do not change it until after you have called the HPMERGEEND intrinsic.

---

*altseq*      **character array (optional)**

Passes an alternate collating sequence. The first element is specified according to the following table, where the sequence comprises the columns and the data comprises the rows:

excess space above nonnumber table—>

	ASCII	EBCDIC	ALTSEQ
ASCII	chr(255)	chr(2)	chr(0)
EBCDIC	chr(1)	chr(255)	undefined

The second element specifies one fewer than the total number of characters in the collating sequence.

If the first element is `CHR(0)`, the remaining array elements comprise the actual collating sequence responsible for the particular MERGE/XL operation.

The defaults are: element 1 = `CHR(255)`, element 2 = `CHR(255)`

*keycompare*      **procedure (reserved)**

Do not specify, but maintain parameter position.

*errorproc*      **procedure (reserved)**

Do not specify, but maintain parameter position.

*statistics*      **32-bit signed integer array (optional)**

Returns information on the MERGE/XL operation:

Value	Meaning
1	Number of input files
2	Number of merged records
3	Number of bytes used for the MERGE/XL work area
4	CPU time in milliseconds
5	Elapsed time in milliseconds

## HPMERGEINIT

*memsize*      **32-bit signed integer by value (reserved)**

Do not specify, but maintain parameter position.

*charseq*      **32-bit signed integer array (optional)**

Passes language information. Set the first element to 1 and the second element to the language ID of the native language collating sequence to be used for keys of type character (*keys* parameter equal to 10).

### Related Information

Intrinsics      Hewlett-Packard SORT, Hewlett-Packard MERGE

Commands      None

Manuals      *SORT-MERGE/XL Programmer's Guide* (32650-90080)



## HPMERGEOUTPUT

NM callable only.

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

### Syntax

```

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

### Parameters

- status*      **32-bit signed integer by reference (optional)**
- Returns the status of the HPMERGEOUTPUT call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.
- Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.
- Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMERGEOUTPUT is 196.
- buffer*      **character array (optional)**
- Returns the next output record. The record format is determined by the value of the *keysonly* parameter in the HPMERGEINIT intrinsic.
- length*      **32-bit signed integer by reference (optional)**
- Returns the length (in bytes) of the *buffer* parameter value. Returns a *length* value of -1 when no more records remain.

### Related Information

- |            |  |
|------------|--|
| Intrinsics | Hewlett-Packard SORT, Hewlett-Packard MERGE            |
| Commands   | None   |
| Manuals    | <i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080). |

---

## HPMERGESTAT

NM callable only.

Prints MERGE/XL statistics on \$STDLIST.

### Syntax

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

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPMERGE<sub>STAT</sub> call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMERGE<sub>STAT</sub> is 196.

*statistics*

**32-bit signed integer array (optional)**

Contains the following statistical information about the MERGE/XL operation:

- Number of input files
- Number of merged records
- Number of bytes used for the MERGE/XL work area
- CPU time in milliseconds
- Elapsed time in milliseconds

This parameter is returned from the HPMERGE<sub>END</sub> intrinsic.

---

### Note

The *statistics* array is the same array that was passed to the HPMERGE<sub>INIT</sub> and HPMERGE<sub>END</sub> intrinsics. It must be referenced by the same name.

---

**Operation Notes**

Call `HPMERGESTAT` after you have called the `HPMERGEEND` intrinsic. A call to `HPMERGESTAT` is valid only if you supplied the *statistics* parameter in the calls to the `HPMERGEINIT` and `HPMERGEEND` intrinsics. The same array is then supplied to `HPMERGESTAT`.

**Related Information**

Intrinsics	Hewlett-Packard <code>SORT</code> , Hewlett-Packard <code>MERGE</code>
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## HPMERGETITLE

NM callable only.

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

### Syntax

I32 HPMERGETITLE( <i>status</i> );
---------------------------------------

### Parameters

*status*      **32-bit signed integer by reference (optional)**

Returns the status of the HPMERGETITLE call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMERGETITLE is 196.

### Operation Notes

HPMERGETITLE prints the date and time produced by the DATELINE intrinsic. You can call HPMERGETITLE from the program at any time after you have declared the system intrinsics.

### Related Information

Intrinsics	Hewlett-Packard SORT, Hewlett-Packard MERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080).

---

## HPMYFILE

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

### Syntax

CA	I32	I32	CA
HPMYFILE( <i>formaldesig</i> , <i>status</i> , <i>length</i> , <i>myfile</i>			
	I32	I32	
<i>myfile_length</i> , <i>myfile_syntax</i> );			

### Parameters

*formaldesig* **character array (required)**

Returns the fully qualified MPE syntax file name of the program or XL that called HPMYFILE.

The *formaldesig* parameter must be at least 28 bytes in length in order to contain the longest possible MPE syntax file name with delimiters. The lockword is not returned. The first and last characters of the returned value are blanks that act as delimiters.

*status* **32-bit signed integer by reference (optional)**

HPMYFILE returns a NM Loader 128 error when the first library name is not represented using the syntax that the user requires. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMYFILE is 104.

---

### Note

Since HPMYFILE can return information on the success of its execution in the *status* parameter, it is good programming practice to specify this parameter and check its values after the intrinsic call. If an error or warning condition is encountered and you did not specify the *status* parameter, HPMYFILE causes the calling process to abort.

---

*length* **32-bit signed integer by reference (optional)**

## HPMYFILE

Returns the length (in bytes) of the file name returned in the *formaldesig* parameter (including the two blanks that act as delimiters). A zero returned indicates that no file name is returned.

*myfile* **character array (optional)**

Returns the absolute pathname or fully qualified MPE syntax name of the program or XL that called HPMYFILE.

The *myfile* parameter contains the name with delimiters. The first and last characters of the returned value are blanks that act as delimiters.

*myfile\_length* **32-bit signed integer by reference (optional)**

On input, the *myfile\_length* parameter specifies the maximum *myfile* length in bytes. On output, *myfile\_length* returns the actual length of *myfile*, including the surrounding delimiters.

If *myfile\_length* is not correct upon input, variables allocated near *myfile* can be overwritten, or an error could occur. If the *myfile\_length* parameter is not specified, HPMYFILE determines if the *myfile* parameter is long enough to hold HPMYFILE to be returned.

Note that a value is not returned in *myfile\_length* if you do not specify the *myfile* parameter.

*myfile\_syntax* **32-bit signed integer by reference (optional)**

On input, the *myfile\_syntax* parameter specifies the syntax that is used when HPMYFILE returns name in *myfile*.

On output, the actual syntax is returned in *myfile\_syntax*. This is only useful when MPE-escaped syntax is used. The name can be either a MPE syntax name or a HFS syntax name.

The possible syntaxes are:

- |   |                    |
|---|--------------------|
| 0 | MPE-escaped syntax |
| 1 | MPE-only syntax    |
| 2 | HFS syntax         |

The default is MPE-only syntax. Note that if the *myfile* parameter is not specified, no value is returned in *myfile\_syntax*.

## Operation Notes

You can pass this file name to the HPGETPROCPLABEL intrinsic in the *firstfile* parameter. HPGETPROCPLABEL searches the files in the binding sequence of its calling process for a procedure, beginning with the file returned by HPMYFILE.

## Related Information

Intrinsics      None

Commands      None

Manuals      *Resource Management Programmer's Guide* (32650-90024).

## HPMYPROGRAM

NM callable only.

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

### Syntax

CA	I32	I32
HPMYPROGRAM( <i>formaldesig</i> , <i>status</i> , <i>length</i>		
CA	I32	I32
<i>myprogram</i> , <i>myprogram_length</i> , <i>myprogram_syntax</i> );		

### Parameters

*formaldesig* **character array (required)**

Returns the fully qualified MPE syntax name of the program being executed by the calling process.

The *formaldesig* parameter must be at least 28 bytes in length in order to contain the longest possible MPE syntax file name with delimiters. The lockword is not returned. The first and last characters of the returned value are blanks that act as delimiters.

*status* **32-bit signed integer by reference (optional)**

HPMYPROGRAM returns a NM Loader error 128 when the program name is not represented using the syntax that the user specifies.

Returns the status of the HPMYPROGRAM intrinsic call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPMYPROGRAM is 104.

## HPMYPROGRAM

<i>length</i>	<b>32-bit signed integer by reference (optional)</b> Returns the length (in bytes) of the file name returned in the <i>formaldesig</i> parameter (including the two blanks that act as delimiters). A zero returned indicates that no file name is returned.						
<i>myprogram</i>	<b>character array (optional)</b> Returns the absolute pathname or fully qualified MPE syntax name of the program being executed by the calling process.  The <i>myprogram</i> parameter contains the name with delimiters. The first and last characters of the returned value are blanks that act as delimiters.						
<i>myprogram_length</i>	<b>32-bit signed integer by reference (optional)</b> On input, the <i>myprogram_length</i> parameter specifies the maximum <i>myprogram</i> length in bytes. On output, <i>myprogram_length</i> returns the actual length of <i>myprogram</i> , including the surrounding delimiters.  If <i>myprogram_length</i> is not correct upon input, variables allocated near <i>myprogram</i> can be overwritten, or an error could occur. If the <i>myprogram_length</i> parameter is not specified, HPMYPROGRAM determines if the <i>myprogram</i> parameter is long enough to hold HPMYPROGRAM to be returned.  Note that a value is not returned in <i>myprogram_length</i> if you do not specify the <i>myprogram</i> parameter.						
<i>myprogram_syntax</i>	<b>32-bit signed integer by reference (optional)</b> On input, the <i>myprogram_syntax</i> parameter specifies the syntax that is used when HPMYPROGRAM returns the name in <i>myprogram</i> . On output, the actual syntax is returned in <i>myprogram_syntax</i> . This is only useful when MPE-escaped syntax is used. The name can be either a MPE syntax name or a HFS syntax name.  The possible syntaxes are: <table><tr><td>0</td><td>MPE-escaped syntax</td></tr><tr><td>1</td><td>MPE-only syntax</td></tr><tr><td>2</td><td>HFS syntax</td></tr></table> The default is MPE-only syntax. Note that if the <i>myprogram</i> parameter is not specified, no value is returned in <i>myprogram_syntax</i> .	0	MPE-escaped syntax	1	MPE-only syntax	2	HFS syntax
0	MPE-escaped syntax						
1	MPE-only syntax						
2	HFS syntax						

### Operation Notes

The HPMYPROGRAM intrinsic returns the fully qualified file name of the program being executed by this process. You can pass this name to the HPGETPROCPLABEL intrinsic in the *firstfile* parameter. HPGETPROCPLABEL searches the files in the binding sequence for a procedure, beginning with the program file. The program file is the first file in the binding sequence of the calling process.



**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Resource Management Programmer's Guide</i> (32650-90024)

---

## HPPIPE

Creates a pipe file type object defined as device type of streams, file type of pipes, and record format of byte streams.

### Syntax

I32            I32 I32 HPPIPE( <i>read_fd</i> , <i>write_fd</i> , <i>status</i> )
--

### Parameters

- read\_fd*            **32-bit signed integer by reference (required)**  
Returns the *read file descriptor* for the pipe. Attempts to write to a *read file descriptor* results in an error.
- write\_fd*            **32-bit signed integer by reference (required)**  
Returns the *write file descriptor* for the pipe. Attempts to read from a *write file descriptor* results in an error.
- status*            **32-bit signed integer by reference (optional)**  
Returns the *status* of the HPPIPE call. If no errors or warnings are encountered, status returns 32-bits of zero. If errors or warnings are encountered, status is interpreted as two 16-bit fields:
- bits (0:16):        *comprise status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition.
- bits (16:32):        *comprise status.subsys*. The value represents the subsystem that sets the status information. The subsystem identifier for HPPIPE is the file system subsystem 143.

---

**Caution**        If an error or warning is encountered and the *status* parameter was not specified, HPPIPE causes the calling process to abort.

---

The following *status.info* values can be returned by HPPPIPE:

**Table 4-21.**

Status Code	Meaning
-461	Maximum open files for operating system exceeded.
-18	Bounds violation on <i>read_fd</i> parameter.
-18	Bounds violation on <i>write_fd</i> parameter.
-80	An internal error occurred.
-81	The operation failed due to a lack of system resources.
-179	The user lacks the ability to create an object.

### Operation Notes

Enables creation of a pipe, which is a nameless object that can only be reached from members of its process's family. Returns two file numbers to the calling process that represent the write and read ends of the pipe. Use the **FWRITE** and **FREAD** intrinsics to read and write to the pipe. Data is returned in a first in, first out manner.

### Related Information

Intrinsics	<b>FCLOSE</b> , <b>FCONTROL</b> , <b>FFILEINFO</b> and <b>FCHECK</b>
Commands	None
Manuals	<i>Resource Management Programmer's Guide</i> (32650-90024)

---

## HPRESETDUMP

NM callable only.

Disarms the system debugger call from a process abort.

### Syntax

I32 HPRESETDUMP( <i>status</i> );
--------------------------------------

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPRESETDUMP call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPRESETDUMP is 165.

### Related Information

Intrinsics      HPSETDUMP , SETDUMP , RESETDUMP

Commands        RESETDUMP

Manuals         *MPE/iX System Debug Reference Manual* (32650-90013)

## HPSELECT

Determine whether a file or group of files is ready to perform reads, writes, or if it has an exception condition, and whether to block on one or more of these events if the condition is not currently true.

### Syntax

I16	IV	CA	CA
<i>numselect</i> := HPSELECT( <i>numfiles</i> , <i>readmask</i> , <i>writemask</i> ,			
	CA	R	I32
<i>exceptionmask</i> , <i>timeout</i> , <i>status</i> )			

### Functional Return

*numselect*      **16-bit signed integer (assigned functional return)**

Returns a value representing the number of files that have selected true. The following values are returned:

Value	Meaning
<i>n</i>	The number of files that have selected true.
0	If a timeout value was assigned and the select “timed out”.
-1	If an error occurred in the call, an internal error occurred, or if a soft interrupt occurred before any selected events or before the time limit expired.

### Parameters

*numfiles*      **integer by value (required)**

Passes the number of files (number of files -1) whose bits will be checked by HPSELECT.

Note that this parameter does not represent the number of files where a bit in one of the masks is being set. It represents the highest file number (minus 1) that will be examined by HPSELECT.

## HPSELECT

*readmask*

### **character array by reference (optional)**

Specifies which file numbers will be examined to see if the file that is ready for reading should be passed. The least significant bit of each 32-bit word represents the lowest file number in the word. The first word contains the bits for the first 32 files. Subsequent words contain the bits for additional groups of 32 files.

For each file number less than *numfiles*, the corresponding bit in the mask is set at return if the bit was set at entry and the file is ready for reading.

Bits representing file numbers greater than *numfiles* (minus 1) are ignored. The buffer containing the bit mask only needs to be as large as the number of 32-bit words required to contain the bits for *numfiles* files.

If a timeout is specified and the time limit has elapsed, the mask is cleared on return.

Default: No files are checked for reading.

*writemask*

### **character array by reference (optional)**

Specifies which file numbers are to be examined to see if the file is ready for writing. The last significant bit of each 32-bit word represents the lowest file number in the word. The first word contains the bits for the first 32 files. Subsequent words contain the bits for additional groups of 32 files.

For each file number less than *numfiles*, the corresponding bit in the mask is set at return if the bit was set at entry and the file is ready for writing.

Bits representing file numbers greater than *numfiles* (minus 1) are ignored. The buffer containing the bit mask only needs to be as large as the number of 32-bit words required to contain the bits for *numfiles* files.

If a timeout is specified and the time limit has elapsed, the mask is cleared on return.

Default: No files are checked for writing.

*exceptionmask* **character array by reference (optional)**

Specifies which file numbers are to be examined to see if the file has detected an exception condition. The last significant bit of each 32-bit word represents the lowest file number in the word. The first word contains the bits for the first 32 files. Subsequent words contain the bits for additional groups of 32 files.

For each file number less than *numfiles*, the corresponding bit in the mask is set at return if the bit was set at entry and the file has detected an exception condition.

Bits representing file numbers greater than *numfiles* (minus 1) are ignored. The buffer containing the bit mask only needs to be as large as the number of 32-bit words required to contain the bits for *numfiles* files.

If a timeout is specified and the time limit has elapsed, the mask is cleared on return.

Default: No files are checked for exception conditions.

*timeout* **record by reference (optional)**

Specifies a maximum interval to wait for the selection to complete. If a *timeout* value of 0 is supplied, HPSELECT checks for the specified conditions, sets the return mask, and returns without blocking even if none of the specified conditions are true. If a *timeout* value is not specified, HPSELECT waits until an event causes one of the masks to be returned with a non-zero value.

The *timeout* value is specified by a 64-bit record containing the following two fields:

32-bit unsigned integer =       seconds

32-bit integer =                   microseconds

Values up to 2\*\*32 seconds can be specified.

Default: HPSELECT blocks until one of the specified conditions is true.

## HPSELECT

*status*

### 32-bit integer by reference (optional)

Returns the status of the HPSELECT call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSELECT is 143.

Possible error values of *status.info* are:

Value	Meaning
-9	Illegal parameter - the value of <i>numfiles</i> is negative or an invalid <i>timeout</i> value was specified.
-18	Bounds violation - one or more parameters point to an invalid address or to an address the user does not have access to.
-30	Invalid file reference - one or more of the bit masks specified an invalid file number.
-90	Timeout - the time limit expired before any selected events occurred.
-91	Soft interrupt - a soft interrupt occurred before any selected events occurred or before the time limit expired.
No status	If status is not passed in, no status is returned.

---

### Note

HPSELECT will not ESCAPE if an error is detected and the *status* parameter was not supplied. Errors are also indicated by a -1 in the *numselect* parameter (functional return). For assurance of a successful completion, check the functional return.

The only file type supported by HPSELECT is Net IPC. For all other file types, HPSELECT always returns TRUE.

---

### Related Information

Intrinsics	IOWAIT
Commands	None
Manuals	None



## HPSETCCODE

NM callable only.

Sets the condition code for the calling process.

### Syntax

```

I32V
HPSETCCODE(ccodevalue);
```

### Parameters

*ccodevalue*    **32-bit signed integer by value (required)**

Sets the condition code for the calling process.

The valid values are:

Value	Meaning
0	Condition Code Greater Than (CCG). A special condition occurred but may not have affected the execution of the request.
1	Condition Code Less Than (CCL). The request was not granted because an error condition occurred.
2	Condition Code Equal (CCE). The request was granted.

Any other values are converted to 1 (CCL).

### Operation Notes

The condition code is implemented as two bits associated with each process. From the condition code value, basic information about what happened during execution of an intrinsic can be learned.

Although Pascal\XL has a built-in function that lets you check the condition code, it does not allow you to set it. If you write procedures and want to return information through a condition code, then use the HPSETCCODE intrinsic. It is the only mechanism for setting the condition code. Only source code compatible routines (implemented in native mode or called through native mode switch stubs) should call HPSETCCODE.

The HPSETCCODE intrinsic is the only mechanism available for setting the condition code in the native mode environment.

## HPSETCCODE

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)

---

## HPSETDUMP

NM callable only.

Arms the system debugger call from a process abort.

### Syntax

I32   CA HPSETDUMP( <i>status</i> , <i>cmdstr</i> );
---

### Parameters

*status*

**32-bit signed integer (optional)**

Returns the status of the HPSETDUMP call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSETDUMP is 165.

*cmdstr*

**character array (optional)**

An array of characters of up to 255 characters containing the system debugger commands. The first character in the buffer is recognized as the delimiter. The last character in the command string must be immediately followed by that same delimiter.

## HPSETDUMP

### Operation Notes

The process enabling the system debugger can be the current process or any child of the current process created after the intrinsic call. The commands are contained in a character array, left-justified, with a delimiting character as the first and last character of the command string.

Before a process aborts, Debug is called to execute the commands in *cmdstr*. Any command that attempts to obtain user input causes an error when executed by Debug. If the process that aborts is being run from a job, the process terminates after executing the command string. If the process is being run from a session, then after the specified command string has been executed, Debug stops to accept interactive commands with I/O performed at the user terminal if the following conditions are met: the abort did not occur while in system code; and the process entered the abort code through a native mode interrupt (typically caused by arithmetic and code related traps).

Once Debug accepts interactive input, the user is free to enter any DEBUG command. The user may resume the process or terminate it with the Debug CONTINUE command.

If the abort is caused by a stack overflow, a stack trace is printed and the process is terminated immediately and the command string is not executed.

### Related Information

Intrinsics      HPDEBUG, HPRESETDUMP, RESETDUMP, SETDUMP, STACKDUMP

Commands      DEBUG, SETDUMP

Manuals      *MPE/iX System Debug Reference Manual (32650-90013)*

---

## HPSORTEND

NM callable only.

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

### Syntax

I32    I32A HPSORTEND( <i>status,statistics</i> );
---

---

**Note**            HPSORTEND must be called to terminate SORT/XL and release the work space.

---

### Parameters

*status*            **32-bit signed integer by reference (optional)**

Returns the status of the HPSORTEND call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTEND is 195.

*statistics*        **32-bit signed integer array (optional)**

Returns SORT/XL operation information:

Value	Meaning
1	Number of records sorted
2	Number of intermediate passes
3	Number of bytes used for the Hewlett-Packard SORT program work area
4	Number of compares
5	CPU time used in milliseconds
6	Elapsed time in milliseconds

---

**Note**            The *statistics* array should be the same array that was passed to the HPSORTINIT intrinsic. It must be referenced by the same name.

---

## **HPSORTEND**

### **Operation Notes**

HPSORTEND initiates the merge phase of the Hewlett-Packard SORT program and writes to the output file if the HPSORTOUTPUT intrinsic is not used. HPSORTEND is required if you called HPSORTINIT. Call HPSORTEND either after all calls to the output file are completed by the HPSORTINIT intrinsic, or after all calls to the HPSORTOUTPUT intrinsic are completed. HPSORTEND must be called in the same process as the HPSORTINIT intrinsic.

### **Related Information**

Intrinsics        HPSORT, HPMERGE

Commands        None

Manuals         *SORT-MERGE/XL Programmer's Guide* (32650-90080).

## HPSORTERRORMESS

NM callable only.

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

### Syntax

I32	CA	I32
HPSORTERRORMESS( <i>status,message,length</i> );		

### Parameters

- status*      **32-bit signed integer by reference (optional)**
- Passes the status of a previous HPSORT intrinsic call that failed. The *status* value passed in is used to locate the appropriate message catalog entry associated with the *status* value. If no errors or warnings were encountered during the execution of HPSORTERRORMESS, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.
- Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.
- Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTERRORMESS is 195.
- message*      **character array (optional)**
- Returns the text of the error message. This parameter may be up to 80 characters long.
- length*      **32-bit signed integer by reference (optional)**
- Returns the actual length of the *message* parameter in characters.

### Related Information

- |            |  |
|------------|--|
| Intrinsics | HPSORT, HPMERGE  |
| Commands   | None   |
| Manuals    | <i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080). |

---

## HPSORTINIT

NM callable only.

Initializes the SORT/XL subsystem.

### Syntax

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

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPSORTINIT call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTINIT is 195.

*inputfiles*

**32-bit signed integer array (optional)**

Passes the file identification numbers of the input files to be sorted. The last element of this array should be set to zero. If the files are opened with either the NOBUF or MR (multirecord) access option (*aoption*), the Hewlett-Packard SORT program performs the buffering and blocking\deblocking. \$NULL is not a valid input file.

If this parameter is not used, there must be a call to HPSORTINPUT to supply records to the Hewlett-Packard SORT program.



*outputfiles*     **32-bit signed integer array (optional)**

Passes the file identification number of the output file. The second array element must be a zero, indicating the end of the array. If the files are opened with either the **NOBUF** or **MR** (multirecord) access option (*aoption*), the SORT/XL program performs the buffering and blocking\deblocking.

If you do not supply this parameter, you must call HPSORTOUTPUT to retrieve the sorted records.

*outputoption*     **32-bit signed integer by value (optional)**

Passes a value determining the format of the output records:

Value	Meaning
0	Output record the same as input record (default)
1	Output record, a 32-bit integer output in binary format, value is the logical (relative) record number of the record
2	Output record contains only the key fields, concatenated together with the major keys on the left, followed by the remaining keys
3	Output record is the logical record number (in binary format) followed by the key fields

*reclength*     **32-bit signed integer by value (optional)**

Maximum length of a record in bytes. If you do not specify *reclength*, the record length is taken from the first file specified in the *inputfiles* parameter. If you do not specify *inputfiles*, you must specify this parameter.

*numrecs*     **32-bit signed integer by value (optional)**

Upper bound to the number of records to be sorted. Use *numrecs* only if all input records do not reside on disk. *numrecs* may be used to determine the size of the temporary scratch files that SORT builds. The size of the scratch file(s) that SORT uses is calculated in the following way:

If all records are on disk, the size of the scratch file number is derived from the current EOF value of the input file and any value supplied for this parameter is ignored. (If there are multiple input files, then the sum of their EOFs is used).

If all files to be sorted do not reside on disk, or if you do not specify any input files at all, and you supply this parameter, *numrecs* will determine the size of the scratch file(s). Otherwise if you do not supply this parameter, a default value of 10,000 records per tape file is used.

*numkeys*     **32-bit signed integer by value (optional)**

Number of keys used during the comparison of records. This parameter can be either equal to or greater than one. If you specify the *numkeys* parameter, you must also specify the *keys* parameter. Together, *numkeys* and *keys* describe the way records are sorted.

*keys*     **32-bit integer array (optional)**

## HPSORTINIT

Passes information about the keys used during the comparison of records. If you specify the *keys* parameter, you must specify the *numkeys* parameter. Together, *keys* and *numkeys* describe the way records are sorted.

For each key, there are four entries in the array. The first element gives the position of the first byte of the key within the input record. The second element gives the number of bytes in the key. The third element indicates the type of data:

Value	Meaning
0	Byte type (ASCII or EBCDIC)
1	Twos complement (including 16-bit and 32-bit signed integer)
2	Hewlett-Packard 3000 floating-point format (including real and long)
3	IEEE standard floating-point format; (32-bit, 64-bit, and 128-bit values)
4	Packed decimal with odd number of digits
5	Packed decimal with even number of digits
6	Display trailing sign
7	Display leading sign
8	Display trailing sign separate
9	Display leading sign separate
10	Character
11	Reserved for the operating system
12	Short floating-point decimal
13	Floating-point decimal

The fourth element gives the ordering sequence of the records:

Value	Meaning
0	Ascending
1	Descending

---

**Note** The integrity of the *keys* array must be maintained throughout the SORT/XL operation. Do not change it until after you have called the HPSORTEND intrinsic.

---

*altseq*      **character array (optional)**

Passes an alternate collating sequence. The first element is specified according to the following table, where the sequence comprises the columns and the data comprises the rows:

	ASCII	EBCDIC	ALTSEQ
ASCII	chr(255)	chr(2)	chr(0)
EBCDIC	chr(1)	chr(255)	undefined

The second element specifies one fewer than the total number of characters in the collating sequence.

If the first element is CHR(0), the remaining array elements comprise the actual collating sequence responsible for the particular Hewlett-Packard SORT operation.

Default: element 1 = CHR(255),  
element 2 = CHR(255)

*keycompare*      **procedure (reserved)**

Do not specify this parameter, but maintain parameter position.

*errorproc*      **procedure (reserved)**

Do not specify this parameter, but maintain parameter position.

## HPSORTINIT

*statistics*      **32-bit signed integer array (optional)**

Returns information on the SORT/XL operation:

Value	Meaning
1	Number of records sorted
2	Number of intermediate passes
3	Number of bytes in the SORT/XL program work area
4	Number of compares
5	CPU time used in milliseconds
6	Elapsed time in milliseconds

*memsize*      **32-bit signed integer by value (reserved)**

Do not specify, but maintain parameter position.

*charseq*      *32-bit signed integer array (optional)*

Passes language information. Set the first element to 1 and the second element to the language ID of the native language collating sequence to be used for keys of type character (*keys* parameter equal to 10).

### Related Information

Intrinsics      HPSORT, HPMERGE

Commands      None

Manuals      *SORT-MERGE/XL Programmer's Guide* (32650-90080).

## HPSORTINPUT

NM callable only.

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

### Syntax

```

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

### Parameters

<i>status</i>	<p><b>32-bit signed integer by reference (optional)</b></p> <p>Returns the status of the HPSORTINPUT call. If no errors or warnings are encountered, <i>status</i> returns 32 bits of zero. If errors or warnings are encountered, <i>status</i> is interpreted as two 16-bit fields.</p> <p>Bits (0:16) comprise <i>status.info</i>. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the <i>MPE/iX Error Message Manual Volumes 1, 2 and 3</i> (32650-90066, 32650-90152, and 32650-90368) for a description of its value.</p> <p>Bits (16:32) comprise <i>status.subsys</i>. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTINPUT is 195.</p>
<i>buffer</i>	<p><b>character array (optional)</b></p> <p>Passes the next input record.</p>
<i>length</i>	<p><b>32-bit signed integer by value (optional)</b></p> <p>Length of the <i>buffer</i> parameter in bytes. This value should be long enough to contain all specified keys, but not longer than the record size specified in the <i>reclength</i> parameter of the HPSORTINIT intrinsic.</p>

### Operation Notes

Provides an alternative method for supplying input records to SORT/XL. Use this intrinsic only if the *inputfiles* parameter in a call to the HPSORTINIT intrinsic was not used. A call to HPSORTINPUT follows a call to HPSORTINIT and precedes calls to HPSORTOUTPUT and HPSORTEND.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080).

---

## HPSORTOUTPUT

NM callable only.

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

### Syntax

I32    CA    I32 HPSORTOUTPUT( <i>status</i> , <i>buffer</i> , <i>length</i> );
--

### Parameters

- status*      **32-bit signed integer by reference (optional)**
- Returns the status of the HPSORTOUTPUT call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.
- Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.
- Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTOUTPUT is 195.
- buffer*      **character array (optional)**
- Returns the next output record. The format of this record is as specified in the *outputoption* parameter of the HPSORTINIT intrinsic.
- length*      **32-bit signed integer by reference (optional)**
- Returns the length of the *record* parameter value. When no more records remain, the value of this parameter is set to -1.

### Operation Notes

Provides an alternative method for retrieving output records from SORT/XL. HPSORTOUTPUT puts each output record from SORT/XL into the array specified by the *buffer* parameter. If the HPSORTINPUT intrinsic was used, HPSORTOUTPUT signals the end of the input process; call HPSORTOUTPUT only after HPSORTINPUT has passed all records. Use HPSORTOUTPUT only if the *outputfiles* parameter was not specified in the call to HPSORTINIT. A call to HPSORTOUTPUT always precedes a call to HPSORTEND.

**Related Information**

Intrinsics      HPSORT, HPMERGE

Commands      None

Manuals      *SORT-MERGE/XL Programmer's Guide* (32650-90080).

---

## HPSORTSTAT

NM callable only.

Prints the SORT/XL statistics on \$STDLIST.

### Syntax

I32    I32A HPSORTSTAT( <i>status,statistics</i> );
--

### Parameters

*status*

**32-bit signed integer by reference (optional)**

Returns the status of the HPSORTSTAT call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTSTAT is 195.

*statistics*

**32-bit signed integer array (optional)**

Contains the following statistical information about the SORT/XL operation:

- Number of sorted records
- Number of intermediate passes
- Number of bytes used for the work area
- Number of compares
- CPU time in milliseconds
- Elapsed time in milliseconds

This parameter is returned from the HPSORTEND intrinsic.

---

### Note

The *statistics* array should be the same array that was passed to the HPSORTINIT and HPSORTEND intrinsics. It must be referenced by the same name.

---



**Operation Notes**

Call HPSORTSTAT after you have called the HPSORTEND intrinsic. A call to HPSORTSTAT is valid only if the *statistics* parameter is specified to the HPSORTINIT and HPSORTEND intrinsics. The same array is supplied to HPSORTSTAT.

**Related Information**

Intrinsics        HPSORT, HPMERGE

Commands        None

Manuals         *SORT-MERGE/XL Programmer's Guide* (32650-90080).

---

## 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.

### Syntax

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

### Parameters

*status*      **32-bit signed integer by reference (optional)**

Returns the status of the HPSORTTITLE call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSORTTITLE is 195.

### Operation Notes

Call HPSORTTITLE any time after declaring the system intrinsics.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## HPSWITCHTOCM

NM callable only.

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

### Syntax

```

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

### Parameters

*proc*            **record (required)**

Passes the target CM procedure identifier, which you can specify either by a library to search and an ASCII name of up to 16 characters or by a CM label (obtained from the HPLOADCMPROCEDURE or LOADPROC intrinsic).

The structure of the *proc* record varies, depending on how the target procedure is identified. All variants have a *p\_proc\_id\_type* field.

*method*        **32-bit signed integer by value (required)**

Passes a value indicating whether the CM target procedure runs in normal, split-stack, or no-copy mode. Valid values are:

Value	Meaning
0	Normal (non-split-stack)
1	Split-stack
2	No-copy

---

### Note

If all parameters are within the CM stack or the length of the reference parameters are less than the threshold, use the normal value. If the reference parameters are outside of the CM stack (split-stack), wrap the reference parameters in an extra data segment if the reference parameters are outside of the CM stack (split-stack method). The no-copy method is the same as the split-stack method, except that the threshold check is omitted.

---

## HPSWITCHTOCM

<i>numparms</i>	<b>32-bit signed integer by value (required)</b> The number of parameters you are passing to the CM target procedure.
<i>parms</i>	<b>array of records (required)</b> Passes descriptions of each parameter being passed to the CM target procedure. Each parameter is located and described by a record in this array.
<i>fretlen</i>	<b>32-bit signed integer by value (optional)</b> The length in bytes of the optional functional return value.
<i>fretval</i>	<b>record by value (optional)</b> NM pointer to the beginning of the area to which the optional functional return value is returned.
<i>condcode</i>	<b>16-bit signed integer by reference (optional)</b> Returns the condition code that the target CM procedure returns. Valid values are in the range 0..2.

---

**Note** If a call to HPSWITCHTOCM fails, the value of the *condcode* parameter becomes undefined. Before attempting to use the *condcode*, first check the value of the *status* parameter.

---

<i>status</i>	<b>32-bit signed integer by reference (optional)</b> Returns the status of the HPSWITCHTOCM intrinsic call. If no errors or warnings are encountered, <i>status</i> returns 32 bits of zero. If errors or warnings are encountered, <i>status</i> is interpreted as two 16-bit fields.  Bits (0:16) comprise <i>status.info</i> . A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the <i>MPE/iX Error Message Manual Volumes 1, 2 and 3</i> (32650-90066, 32650-90152, and 32650-90368) for a description of its value.  Bits (16:32) comprise <i>status.subsys</i> . The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPSWITCHTOCM is 100.
---------------	---

### Operation Notes

There are differences in data and variable representations between NM and CM. Programs that switch from NM to CM must use the HPSWITCHTOCM intrinsic which provides mixed-mode execution access and performs the necessary parameter translation between the two modes of execution.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)

---

## 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.

### Syntax

CA	I16V	CA	I16V
status:= HPSWTONMNAME( <i>procname</i> , <i>proclen</i> , <i>libname</i> , <i>liblen</i> ,			
I16V	I16	I16	I16V
<i>nparms</i> , <i>arglist</i> , <i>argdesc</i> , <i>func type</i> )			

### Functional Return

*status*           **I32 (assigned functional return)**  
Returns a 32-bit integer indicating the status of the call.

### Parameters

*procname*       **character array by reference (required)**  
Passes the target procedure name. The target procedure must be contained in an executable library (XL). If the value of *procname* is invalid, blank, or does not contain the NM procedure, NL.PUB.SYS is searched.

*proclen*         **16-bit signed integer by value (required)**  
Passes the byte length of the procedure name.

*libname*         **character array by reference (required)**  
Passes the name of the NM library to be searched. If the value of *libname* is invalid, blank or does not contain the NM procedure, NL.PUB.SYS is searched.

---

**Note**           By default, the search for the referenced library occurs in the procedure's group and account. In order to reference a library in some other group and/or account, you must be sure that *libname* is a fully qualified name (*name.group.account*)

---

*liblen*         **16-bit signed integer by value (required)**  
Passes the byte length of the library name.

*nparms*         **16-bit signed integer by value (required)**

## HPSWTONMNAME

Passes the number of parameters to be passed to the target NM procedure. It specifies the length of the *argdesc* array. Account for any hidden parameters (for example, parameters, an extensible parameter list, and so forth). For more information on hidden parameters, refer to the *Hewlett-Packard Pascal/XL Reference Manual* (31502-90002).

*arglist*      **16-bit signed integer array by reference (required)**

Passes the actual parameters to be passed in the NM procedure.

*argdesc*      **16-bit signed integer array by reference (required)**

Passes integer codes describing the parameters held in the *arglist* array. Refer to *Switch Programming Guide* (32650-90014) for code descriptions.

*funcype*      **16-bit signed integer by value (required)**

Passes the data type of the value the target procedure returns if it is a function. If the target is not a function, the value of this parameter is zero. Refer to *Switch Programming Guide* (32650-90014) for possible values.

## Operation Notes

Do not use HPSWTONMNAME to invoke a native mode (NM) system supplied intrinsic. The result of the call may be unpredictable.

NM code can address the entire CM stack, so there is no copying of reference parameters. The length of each parameter is not needed because lengths are implied in the descriptor list.

Switches by name involve high system overhead on the first call per name, but substantially lower overhead on each subsequent call for that name. The HPSWTONMNAME, HPSWITCHTOCM, HPLOADCMPROCEDURE, and HPLOADNMPROC intrinsics perform a hashing function on the name of the other-mode procedure and store the label for that procedure in a system internal hash table. The LOADPROC intrinsic, does not perform hashing and involves high system overhead each time it is called.

The strings supplied as values of the *procname* and *libname* parameters must exactly match the names of the target NM routine and its NM library, respectively.

---

**Note**            By default, the search for the referenced library occurs in the procedure's group and account. In order to reference a library in some other group and/or account, you must be sure that *libname* is a fully qualified name (*name.group.account*)

---

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)

## 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.

### Syntax

```

      U32V  I16V   I16   I16
status :=HPSWTONMPLABEL(proc,nparms,arglist,argdesc,
      I16V
      functype);

```

### Functional Return

*status*           **I32 (assigned functional return)**  
 Returns a 32-bit integer indicating the status of the call.

### Parameters

*proc*           **32-bit unsigned integer by value (required)**  
 Passes the NM plabel of the target procedure name. This plabel is usually obtained by calling the HPLOADNMPROC intrinsic.

*nparms*       **16-bit signed integer by value (required)**  
 Passes the number of parameters to be passed to the target NM procedure. It specifies the length of *argdesc*. Account for any hidden parameters (such as, parameters, an extensible parameter list, and so forth). For more information on hidden parameters, refer to the *Hewlett-Packard Pascal/XL Reference Manual* (31502-90002).

*arglist*       **16-bit signed integer array by reference (required)**  
 Passes the actual parameters to be passed to the NM procedure.

*argdesc*       **16-bit signed integer array by reference (required)**  
 Passes integer codes describing the parameters held in the *arglist* array (that is, byte, word, double, pointer, and so forth).

*functype*      **16-bit signed integer by value (required)**  
 Passes the data type of the value the target procedure returns if it is a function. If the target is not a function, the value of this parameter is zero.

## **HPSWTONMPLABEL**

Refer to the *Switch Programming Guide* (32650-90014) for supported function types.

### **Operation Notes**

Do not use HPSWTONMPLABEL to invoke a native mode (NM) system supplied intrinsic. The result of the call may be unpredictable.

NM code can address the entire CM stack, so there is no copying of reference parameters. The length of each parameter is not required because lengths are implied in the descriptor list.

### **Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)



## HPUNLOADCMPROCEDURE

NM callable only.

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

### Syntax

CA      U8V    I32
HPUNLOADCMPROCEDURE( <i>procname</i> , <i>library</i> , <i>status</i> );

### Parameters

*procname*      **character array (required)**

Passes as ASCII procedure name, left-justified and padded with blanks. The name can have a maximum of 16 characters.

*library*        **8-bit unsigned integer by value (required)**

Passes indicator of the CM segmented library to be searched. The valid values are as follows:

Value	Meaning
0	Search the system SL only (default)
1	Search the logon account SL, then the system SL
2	Search the logon group SL, then the logon account SL, and then the system SL
3	Search the program file's account SL, then the system SL
4	Search the program file's group SL, then the program file's account SL, and then the system SL

## HPUNLOADCMPROCEDURE

*status*            **32-bit signed integer by reference (optional)**

Returns the status of the HPUNLOADCMPROCEDURE intrinsic call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error condition, and a positive value indicates a warning condition. Refer to the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368) for a description of its value.

Bits (16:32) comprise *status.subsys*. The value represented by these bits defines the subsystem that set the status information. The subsystem identifier for HPUNLOADCMPROCEDURE is 105.

### Operation Notes

Procedures are automatically unloaded when your process terminates. Use the HPUNLOADCMPROCEDURE intrinsic to unload before the process terminates.

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)

---

## HPVOLINFO

Returns volume information.

Up to six items of information can be retrieved. They must be specified in pairs (*itemnum*, *item*) and separated by commas

### Syntax

I32	I16	*
HPVOLINFO( <i>status</i> , <i>volspecifiernum</i> , <i>volspecifier</i>		
I16	*	
[, <i>itemnum</i> , <i>item</i> ] [...]);		

### Parameters

*status*      **32-bit integer (optional)**

Returns the status of the HPVOLINFO call. If no errors or warnings are encountered, *status* returns 32 bits of zero. If errors or warnings are encountered, *status* is interpreted as two 16-bit fields.

Bits (0:16) comprise *status.info*. A negative value indicates an error. A positive value indicates a warning.

Bits (16:32) comprise *status.subsys*. This value defines the subsystem that set the status information. On MPE/iX the volume management identification number is 163. On MPE V, this field contains a 0 (zero).

---

### Note

It is good programming practice to specify *status* and check its value after the intrinsic call. If you do not specify *status* and an error condition is encountered, HPVOLINFO causes the calling process to abort.

---

## HPVOLINFO

The error (negative) / warning (positive) values of `status.info` that can be returned from a call to `HPVOLINFO` are given here.

Values marked with an asterisk (\*) are returned by the MPE/iX version of `HPVOLINFO` only.

Value	Meaning
0	No errors or warnings (successful call).
150	Array passed in cannot hold all names—list truncated.
151	File label is unreadable.
152 *	Parts of the set or class are not mounted. Data was gathered on the portion of the set or class that was mounted, so the data may be incomplete.
-150	Invalid <i>itemnum</i> .
-151	Missing <i>itemnum</i> or <i>item</i> (not paired).
-152	Required parameter omitted.
-153	Parameter address out of bounds.
-154	Split stack calls not allowed.
-155	LDEV, volume set/class, volume not mounted.
-156	Invalid volume specifier number.
-157	Invalid volume specifier.
-158	LDEV not a disk LDEV.
-159	Device class not configured for volumes.
-160	Device class does not map into a volume class.
-161	Free space range sizes not in ascending order.
-162	Invalid number of free space ranges specified.
-163	Volume class cannot be specified with system set.
-164	Error while scanning the directory.
-165	Volume label is unreadable.
-166	Disk free space map is bad.
-167	Allocation is disabled for disk free map.
-168	Defective tracks table is unreadable.

<b>Value</b>	<b>Meaning</b>
-169	Disk I/O error.
-170	Virtual memory is only valid on system volumes.
-171	Directory size is not valid for specified volume.
-172	Spoolfiles are valid only on system volumes.
-173	Item number is valid only on MPE/iX systems.
-174	Item number is valid only on MPE V systems.
-175	List length specified is invalid.
-176	Item number / volume specifier number combination is illegal.
-177	Item is not yet implemented.
-178	Volume table is in an inconsistent state.
-179	Unrecognized drive type.
-180 *	Physical I/O error.
-181 *	An unexpected error was detected. Check the error stack to determine the exact error.
-182 *	An error was detected in the label management subsystem. Check the error stack to determine the exact error.
-183 *	Disk access error. Check the error stack to determine the exact error.
-184 *	A mirrored volume that was specified is disabled.
-185 *	The volume set specified is not a volume set on the system.
-186 *	The volume class specified is not present in the volume set specified.
-187 *	The member volume specified is not present in the volume set specified.

## HPVOLINFO

*volspecifiernum*

### 16-bit signed integer by value (optional)

A 16-bit integer indicating which volume specifier is to be used to obtain information from HPVOLINFO. The default value for *volspecifiernum* is 0 (zero) if it is not included by the caller. This parameter is used in conjunction with the *volspecifier* parameter.

The following are the valid volume specifier numbers:

Value	Meaning
0	Volume specifier is ignored. Using 0 (zero) is the same as specifying all the volumes on the system.
1	Volume specifier is the logical device number of the volume for which information is to be obtained.
2	Volume specifier is a volume set name.
3	Volume specifier is a volume set/volume class name pair.
4	Volume specifier is a volume set/volume name pair.
5	Volume specifier is a device class name.

*volspecifier*

### Type varies (optional).

The *volspecifier* is optional when *volspecifiernum* is 0 (zero).

However, for a *volspecifiernum* of 1, 2, 3, 4, or 5, you must specify a *volspecifier* parameter.

On MPE V, you must declare *volspecifier* as a byte array.

The data type of the *volspecifier* depends on the *volspecifiernum*:

<i>volspecifiernum</i>	<i>volspecifier</i>
0	Ignored
1	16-bit signed integer
2, 3, 4, 5	Character array

When 0 (zero) is used as the *volspecifiernum*, the *volspecifier* is ignored. A *volspecifiernum* of 0 (zero) refers to all the volumes on the system. This includes all system and nonsystem volumes.

When 1 is used as the *volspecifiernum*, the *volspecifier* must be an LDEV number that corresponds to a configured and mounted volume. An LDEV number can be any number from 1 through 999.

For character specifiers, the first character determines the delimiter character for the parameter. The delimiter must be printable; it cannot be alphanumeric or one of the special characters “. , :” or “\_” as explained below. For example, to pass the volume set MVSN1, you may use % as the delimiter:

```
%MVSN1%
```

## Operation Notes

- For *volspecifiernum* of 3 or 4, the volume set name and the volume class or volume name must be separated by a colon:

```
%SET:CLASS%
```

A colon cannot be used as a delimiter.

- Since a volume set or class can be fully qualified on MPE V, periods are allowed *in the name*. Because periods and underscores are allowed in an MPE/iX name, these characters cannot be used as delimiters.
- For *volspecifiernum* of 2 or 3 on MPE V (private volumes), a volume set/class name may be partially or fully qualified:

```
VSETNAME.GROUP.ACCT
```

```
CLASSNAM.GROUP.ACCT
```

Each field of the name must consist of one to eight alphanumeric characters. The first character must be a letter. The name may have up to 55 characters:

```
%SETxxxxx.GROUPxxx.ACCOUNTx:CLASSxxx.GROUPxxx.ACCOUNTx%
```

This name has six 8-character fields, four periods, a colon, and two delimiters. When the volume set or class name is not fully qualified, it refers to the volume set or class in the logon group/account.

- On MPE/iX, a valid volume set or class name may contain as many as 32 characters. The first character must be a letter. The rest may be any combination of alphanumeric characters, including the underscore “\_” and the period “.”.
- On MPE/iX, the name of the system volume set is MPEXL\_SYSTEM\_VOLUME\_SET. The system volume set on MPE V does not have a predefined name; thus, the name MPEXL\_SYSTEM\_VOLUME\_SET is used for compatibility with MPE/iX.

On both systems, this is the volume set name returned for items #3 and #12.

- When the system volume set name is used on MPE V, it must not be followed by a class name (Refer to the discussion for *volspecifiernum* 5, below)

On MPE V, group and account names cannot be used to partially or fully qualify the set or volume name when the system volume set name is specified.

- For *volspecifiernum* 4 on MPE V, a volume name consists of as many as eight alphanumeric characters. The first character must be a letter. For private volumes, names may be partially or fully qualified (*vname.group.acct*). Each field of the name must be one to eight alphanumeric characters, the first of which must be a letter. The maximum number of characters allowed is 55.

```
%SETxxxxx.GROUPxxx.ACCOUNTx:NAMExxxx.GROUPxxx.ACCOUNTx%
```

This name has six 8-character fields, four periods, a colon, and two delimiters.

When the volume name is not fully qualified, it refers to the volume name in the logon group and/or account.

## HPVOLINFO

- For *volspecifiernum* 4 on MPE/iX, the volume name consists of any string of one to 16 alphanumeric characters, including the underscore and the period. The first character must be a letter.
- On MPE V, there is no volume class for the system volume set. In order to access a subset of the system volume set, device classes are used.
- On MPE/iX, volume classes exist for both system and nonsystem volume sets. Volume classes take the place of device classes.
- The *volspecifiernum* 5, for items #6 and #7, provides compatibility between MPE/iX and MPE V. Specifier 5 passes the device class of a group of volumes as they apply to the I/O configuration. The maximum number of characters allowed in the string is 8. On MPE V, specifier 5 must refer to a device class that is configured to a group of volumes.
- ON MPE/iX, when a device class name is specified, a configured and mounted volume class with the same name must exist in the system volume set. Data is returned based on this *volume class* in the *system volume set* if it exists. Otherwise, an error results.

*itemnum*      **16-bit signed integer by value (optional).**

This is the cardinal number of the item desired. It specifies which item value is to be returned. See **Item and Itemnum Pairs**.

*item*      **Type varies (optional).**

On MPE V, the item must be declared as a byte array. The actual type of the *item* is specified by the corresponding *itemnum*. See **Item and Itemnum Pairs**.

## Special Considerations

- You do not need special capabilities to use the **HPVOLINFO** intrinsic.
- Split-stack calls are illegal.
- When an error is returned, the values in the *item* parameters are all undefined. It is not guaranteed that the data returned in an *item* is meaningful if any negative status value is returned.
- On MPE V, real values are accepted from the caller and returned to the caller in 64-bit Hewlett-Packard 3000 format. This is the default format for MPE V.
- On MPE/iX, real values are accepted from the caller and returned to the caller in the format that is the default mode of the caller.

If the intrinsic is called from compatibility mode, any real values specified to **HPVOLINFO** must be in 64-bit Hewlett-Packard 3000 format, and any real values returned by **HPVOLINFO** will be in that format.

If the intrinsic is called from native mode, any real values specified to **HPVOLINFO** must be in 64-bit IEEE format, and any real values returned by **HPVOLINFO** will be returned in that format.

- If the system is highly active while this intrinsic is called, some of the *item* values returned may not reflect the expected results. For example, when **HPVOLINFO** is called to return the disk space used by permanent files, a value is returned. But if immediately after the value is returned, a process on the system purges a file (**FDCLOSE** a file with a disposition of 4), the value does not reflect this difference.



- ON MPE V, a set or class is considered to be logically mounted if all of its members are logically mounted (that is, the LMOUNT or MOUNT command was used).
- On MPE/iX, if the volume specifier is a set or class, then at least the master volume must be logically mounted (that is, the volume must be in a master or member state). If the volume specifier is a logical device number, or a volume name, both the volume specified and the master volume (of the set enclosing the volume specified) must be logically mounted.
- If the volume specifier is a logical device number, it must be a member of a set or class that is logically mounted. If the volume specifier is a volume set name, the set must be logically mounted. If the volume specifier is a volume class name, the class must be logically mounted. If the volume specifier is a volume name, that volume must be a member of a set or class that is logically mounted. When the volume specifier is a device class, all of the logical device numbers in the specified device class must be members of logically mounted sets/classes. All the volumes on the system, whether they are logically mounted or not, may be specified by using *volspecifiernum* 0 (zero).
- If the volumes for *volspecifiernum* 1, 2, 3, 4, 5 are not logically mounted as specified above, the HPVOLINFO intrinsic returns a VOLUME NOT MOUNTED error.

If the caller chooses to continue with the same specifier, the LMOUNT command (MPE V) or the VSOPEN command MPE/iX can be used with the COMMAND or HPCICOMMAND intrinsic to logically mount the appropriate volume sets or classes.

If a volume is taken offline while the intrinsic is accessing it, the process hangs until the volume is online again. I/Os cannot complete to the disk when a volume is offline.

- On MPE/iX, mirrored disks maintain identical copies of the same information on two disks. Consequently, the values returned by this intrinsic reflect information from only one of the volumes in a mirror disk pair. When retrieving an LDEV number (item #13) using a volume name that is associated with a mirrored disk pair, only one of the LDEV numbers of the mirrored pair is returned. Which one is returned is random. Be aware that in this situation, the LDEV number returned may be different across system startups. Later use of the LDEV numbers returned for mirrored disks are guaranteed to provide accurate information for the mirrored volume set.
- On MPE/iX, if information is requested for the split *backup* volume set, the information returned is for the split *user* volume.

## HPVOLINFO

### Information Returned

INFORMATION RETURNED	ITEM#
Disk space used by permanent files, in sectors	32, 33
Disk space used by temporary files, in sectors	(reserved: 34, 35)
Drive type	8
File label overhead, in sectors MPE/iX	26, 27
Free space area, largest contiguous, in sectors	42, 43
Free space distribution array	36, 37
Free space distribution, in sectors per range	38, 39
Free space, total, in sectors	40, 41
Logical device number	13
Member volume names, list	7
Member volumes, number of	6
Overhead, directory space, in sectors	24, 25
Overhead, MPE total, in sectors	16, 17
Overhead, transaction management MPE/iX	28, 29
Overhead, transient space, in sectors MPE/iX	18, 19
Sector size, in bytes	9
Spoolfile disk space usage, in sectors	30, 31
Transient space, configured maximum, MPE/iX, in sectors	20, 21
Virtual memory overhead, in sectors MPE V	22, 23
Volume capacity, in sectors	14, 15
Volume class names, list	5
Volume class names, number of	4
Volume name	11
Volume sets, list	3
Volume sets, number of	2
Volume set name	12
Volume type	10

## Item and Itemnum Pairs

All *item* parameters are passed by reference.

### 2            **Number of volume sets**

*Function* = MPE/iX, MPE V: DSTAT

Returns the number of system and nonsystem volume sets that are configured on the system. It returns a 32-bit signed integer containing the number of volumes sets. Zero (0) is the only valid *volspecificrnum*.

May be used with *Item*=3.

### 3            **List of volume set names**

*Function* = MPE/iX, MPE V: DSTAT

Returns a list of all system and nonsystem volume set names mounted on the system. The item must be a character array where the list of set names is returned. The first four bytes of the array are interpreted as a 32-bit integer describing the length of the array. The caller must set this value equal to the maximum number of names that fits into the array being passed. On return, the value has been modified to reflect the actual number of names returned. The remaining bytes are mapped to a list of 32-byte names.

May be used with *Item*=2 to determine the maximum number of names that could be returned.

### 4            **Number of volume classes**

*Function* = MPE/iX: VOLUTIL command

Returns the number of volume classes that a volume or volume set is associated with. Returns a 32-bit signed integer containing the number of volume classes. On MPE/iX, this number includes only classes whose members are logically mounted.

A volume can be associated with more than one volume class. So, when the specifier is 1 or 4, the number returned is the number of volume classes that the volume is a member of. When the specifier is 2, the number returned is the number of volume classes that are a subset of the volume set.

May be used with *Item*=5

5 **List of volume class names**

*Function* = MPE/iX:VOLUTIL

Returns a list of volume class names. On MPE/iX, the list includes only the names of classes whose members are all logically mounted.

The item must be a character array where the list of class names is returned. The first four bytes of the array are interpreted as a 32-bit integer describing the length of the array. The caller must set this value equal to the maximum number of names that fits into the array being passed. The remaining bytes are mapped to a list of 32-byte names. On return, the value has been modified to reflect the actual number of names returned.

A volume can be associated with more than one volume class. So when the specifier is 1 or 4, the list returned is a list of volume classes that the volume is a member of. When the specifier is 2, the list returned is a list of volume classes that are a subset of the volume set.

May be used with *Item*=4 to determine the maximum number of names that could be returned.

6 **Number of member volumes**

*Function* = MPE V: LISTVS MPE/iX: VOLUTIL

Returns the number of member volumes in the specified volume set/class or device class. This item returns a 32-bit signed integer containing the number of member volumes.

When the specifier is 2 or 3, the number of member volumes that make up the volume set or class is returned, including the master volume in the total count. When the specifier is 5, the number of member volumes configured with the passed device class is returned.

For example, if a volume set consists of a master volume, **MASTER**, and two member volumes, **MEMBER1** and **MEMBER2**, and if the specifier used is a volume set name, the number returned is 3.

On MPE/iX you can remove some or all of the volumes in a volume set (excluding the master volume). Therefore, this number depends on the number of volumes you have mounted at the time the intrinsic call is made. Mirrored disks maintain identical copies of the same information on two disks. Therefore, the number returned reflects only one of the volumes in a mirrored disk pair. Which volume is chosen is random.

On MPE V, the concept of master and member volumes refers only to private volumes. Therefore, if the system volume set is specified, the count consists of the number of volumes that are designated as system volumes.

May be used with *Item*=7.

7 **List of member volume names**

*Function* = MPE/iX: VOLUTIL MPE V: LISTVS

Returns the list of names of the member volumes in the specified volume set/class or device class.

The item must be a character array where the list of member volumes is returned. The first four bytes of the array are interpreted as a 32-bit integer describing the length of the array. The caller must set this value equal to the maximum number of names that fits into the array being passed. On return, the value has been modified to reflect the actual number of names returned. The remaining bytes will be mapped to a list of 32-byte names.

Item 6 may be used to determine the maximum number of names that can be returned.

When the specifier is 2 or 3, the list of member volumes that make up the set or class is returned. When the specifier is 5, the list of member volumes configured with the passed device class is returned.

On MPE/iX, you can remove some or all of the volumes in a volume set (excluding the master volume). Therefore, this list depends on the number of volumes you have mounted at the time the intrinsic call is made.

On MPE/iX, a volume class need not include the master volume. As a result, specifier 3 may return a list that does not include the master volume.

On MPE V, the concept of master and member volumes refers only to private volumes. Therefore, if the system volume set is specified, the count consists of the number of volumes that are designated as system volumes.

## HPVOLINFO

- 8           **Drive type**  
*Function = MPE/iX, MPE V: DSTAT*  
Returns the type of the drive specified. The drive type refers to the name of the drive (for example: HP7935, HP7937). The character array that is used must be large enough to contain the longest type string, currently 13 characters.
- 9           **Drive sector size**  
Returns the logical sector size of the specified drive. This item returns a 32-bit signed integer containing the drive's local sector size in bytes.  
Currently, this logical size is 256 bytes. In the future, however, disks may have different physical sector sizes. MPE will map them to system-wide logical sector sizes.
- 10          **Volume type**  
This item returns a 32-bit integer specifying the volume type. Valid types are:  
1 - System volume      2 - Nonsystem volume  
On MPE, there are two types of volume sets: the system volume set and nonsystem volume sets. A volume from the system volume set is considered a system volume; a volume from the nonsystem set is considered a nonsystem volume. A nonsystem set on MPE/iX is equivalent to a private volume set on MPE V.
- 11          **Volume name**  
*Function = MPE V: VINIT with PLABEL*  
Returns the volume name of the specified LDEV. This item is a character array and must be specified with a length of 32 bytes.
- 12          **Volume set name**  
*Function = MPE/iX, MPE V: DSTAT*  
Returns the volume set name corresponding to the passed LDEV. An LDEV can be associated with only one volume set. This item is a character array and must be specified with a length of 32 bytes.
- 13          **Logical device number**  
Returns the logical device number of the specified volume. This item returns a 16-bit signed integer containing the logical device number.  
On MPE/iX, mirrored disks maintain identical copies of the same information on two disks. Information is returned for only one of the disks. Which is returned is random.
- 14 & 15     **Volume capacity**  
*Function = MPE/iX: VOLUTIL command*  
Returns the volume capacity.  
Item 14 returns a 64-bit signed integer containing the volume capacity in sectors. Item 15 returns the capacity as a 64-bit real.

When the specifier is 1 or 4, the volume capacity consists of the capacity of the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the capacity of the volumes that make up the volume set or class are totalled. This total value is returned.

16 &amp; 17

**Total MPE overhead**

Returns the total MPE overhead, which consists of everything on a volume that is not set aside for file space use. This is the volume space used for operating system purposes.

Item 16 returns a 64-bit signed integer containing the total MPE overhead. Item 17 returns the overhead as a 64-bit real.

Files that compose the code for the operating system are not included in this overhead. Some of the space that is considered overhead includes:

MPE V - volume label, virtual memory, directory, defective tracks/sectors table, initial segments, disk coldload information table, volume table, free space map, and channel programs.

MPE/iX - volume label, file label table, directory, volume set information table, free space map, transient space, and transaction management overhead.

A subset of the overhead is returned through *itemnums* 18 through 29:

- 18 & 19 - MPE transient space
- 20 & 21 - MPE/iX configured transient space
- 22 & 23 - MPE V virtual memory
- 24 & 25 - Directory
- 26 & 27 - MPE/iX file label tables
- 28 & 29 - MPE/iX transaction management

When the specifier is 1 or 4, the total MPE overhead consists of the MPE overhead on the volume whose LDEV or volume name was specified. These are subsets of items 16 and 17.

When the specifier is 2 or 3, the MPE overhead on the volumes that make up the set or class is returned.

18 &amp; 19

**MPE/iX transient space overhead**

Returns the MPE/iX transient space overhead. This item is valid only on MPE/iX.

These are subsets of items 16 and 17.

Item 18 returns a 64-bit signed integer containing the transient space overhead in sectors. Item 19 returns this overhead as a 64-bit real.

Transient space overhead consists of volume space that is used for temporary processes, such as stacks, heaps, and the operating system data structure.

When the specifier is 1 or 4, the transient space consists of the transient space on the volume whose LDEV or volume name was specified.

## HPVOLINFO

When the specifier is 2 or 3, the transient space on the volumes that make up the set or class is returned.

20 & 21

### **Configured maximum MPE/iX transient space**

*Function* = MPE/iX: VOLUTIL command

Returns the configured maximum MPE/iX transient space.

These are subsets of items 16 and 17.

Item 20 returns a 64-bit signed integer containing the maximum transient space overhead in sectors. Item 21 returns this overhead as a 64-bit real.

This is volume overhead that is configured for transient space use. It is not necessarily used. This is space configured for stacks, heaps, and operating system structures.

Refer to Items 18 and 19 above.

When the specifier is 1 or 4, the configured maximum transient space consists of the configured transient space on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the configured transient space on the volumes that make up the volume set or class is returned.

22 & 23

### **MPE V virtual memory overhead**

*Function* = MPE V: SYSDUMP virtual memory device allocation portion (SM capability required to use SYSDUMP).

Returns the MPE virtual memory overhead. This is valid only on MPE V.

These are subsets of items 16 and 17.

Item 22 returns a 64-bit signed integer containing the virtual memory overhead in sectors. Item 23 returns this overhead as a 64-bit real.

MPE V virtual memory overhead is the volume space that is allocated for saving segments of main memory temporarily. This amount of configured virtual memory space is equivalent to the amount of used virtual memory space. Virtual memory on MPE V is allocated only on system volumes. There is no virtual memory on private volumes.

When the specifier is 1 or 4, the virtual memory overhead consists of the virtual memory on the volume whose LDEV or volume name was specified.

When the specifier is 2, the only valid specifier is `MPEXL_SYSTEM_VOLUME_SET`. This is because virtual memory is allocated only on system volumes.

Specifier 3 is invalid. This is because virtual memory is valid only on the MPE V system volume set, and because the concept of volume classes does not hold for MPE V system volumes.

24 & 25

### **Directory space overhead**

Returns the directory space overhead.

Item 24 returns a 64-bit signed integer containing the directory space overhead in sectors. Item 25 returns this overhead as a 64-bit real.



These are subsets of items 16 and 17.

Directory space is area on system and nonsystem volumes reserved for accounting information. It consists of the directory space used for permanent files.

When the specifier is 1 or 4, the directory space overhead consists of the directory space on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the directory space overhead on the volumes that make up the volume set or class is returned.

On MPE V, the system directory is allocated on LDEV 1, and for private volumes, the directory is allocated on the master volume of the volume set or class. Specifying a volume set or class gives the same results as specifying the master volume of the set or class.

26 & 27

#### **MPE/iX file label overhead**

Returns the MPE/iX file label overhead. This item is valid only on MPE/iX.

Item 26 returns a 64-bit signed integer containing the file label overhead in sectors. Item 27 returns this overhead as a 64-bit real.

These are subsets of items 16 and 17.

On MPE/iX, each volume has its own label table, which contains file labels and extent descriptors for files that begin on that volume.

When the specifier is 1 or 4, the file label overhead consists of the overhead on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the file label overhead on the volumes that make up the set or class is returned.

28 & 29

#### **MPE/iX transaction management overhead**

Returns the MPE/iX transaction management overhead. This item is valid only on MPE/iX.

Item 28 returns a 64-bit signed integer containing the transaction management overhead in sectors. Item 29 returns this overhead as a 64-bit real.

These are subsets of items 16 and 17.

Transaction management overhead consists of any logging information that is maintained in order to provide file consistence and file recovery.

When the specifier is 1 or 4, the transaction management overhead consists of any logging information kept on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the transaction management overhead on the volumes that make up the set or volume class is returned.

## HPVOLINFO

30 & 31

### **Spoolfile disk space usage**

*Function* = MPE/iX, MPE V: SHOWOUT

Returns the MPE spoolfile disk space usage.

Item 31 returns a 64-bit signed integer containing the spoolfile disk space usage in sectors. Item 32 returns this usage as a 64-bit real.

Spoolfile space consists of the volume space that is used by hidden spoolfiles. Hidden spoolfiles are not part of the permanent file space. This space can be found only on system volumes that are configured with the device class `SP00L`. Spool file space is not allocated on nonsystem volumes.

When the specifier is 1 or 4, the spoolfile space consists of the space used on the volume whose LDEV or volume name was specified.

When the specifier is 2, the volume set name must refer to the system volume set name.

Specifier 3 is valid only on MPE/iX.

32 & 33

### **Disk space used by permanent files**

*Function* = MPE/iX, MPE V:REPORT @.@

Returns the disk space used by permanent files.

Item 32 returns a 64-bit signed integer containing the disk space used by files in sectors. Item 33 returns this space as a 64-bit real.

When the specifier is 1 or 4, the disk space used by files consists of the space used by permanent files on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the disk space used by permanent files on the volumes that make up the volume set or class is returned.

34 & 35

### **Reserved for disk space used by temporary files.**

36 & 37

### **Free space distribution array**

*Function* = MPE/iX: DISCFREE MPE V: FREE5

Returns the free space distribution array. The caller passes an array that specifies a series of ranges. The intrinsic returns the number of free areas whose size is within each of the specified ranges. The array must consist of 64-bit values.

The item passed must be a variable length array of 64-bit numbers (maximum array length of 16 elements).

For item 36, the values must be in integer format. It returns an array of 64-bit integers containing the free space distribution.

For item 37, the values must be in real format. Real values is rounded off. It returns an array of 64-bit reals.

On MPE V, the largest value that may be specified in either format is 2,147,483,646.

The first value is the number of ranges (minimum 2, maximum 16). The remaining values are the lower bounds for the ranges, in ascending order. On return the first element is the number of free areas whose size is smaller than the smallest bound specified. Each remaining element is the number of free space areas whose size is greater than or equal to the bound, and less than the next larger bound. The following example shows a return in six ranges:

Array Index	Values Passed In	Free Area Size Ranges
1	6	1 - 9 contiguous sectors
2	10	10 - 99 contiguous sectors
3	100	100 - 999 contiguous sectors
4	1000	1000 - 9999 contiguous sectors
5	10000	10,000 - 99,999 contiguous sectors
6	100000	100,000+ contiguous sectors

For example, if there are four areas of free space that are between the size of 100 and 999, the third value of the free space distribution array would contain 4 on return.

Notice that the number of ranges specified in the first element of the array example is 6, but only 5 lower bounds are specified, because the smallest lower bound is assumed to be 1 (one).

When the specifier is 1 or 4, the free space distribution array consists of the contiguous free space areas on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the free space distribution array returns the total values from the volumes that make up the volume set or class.

38 &amp; 39

### **Free space distribution sectors per range**

*Function* = MPE/iX: DISCFREE, MPE V: FREE5

Returns the free space distribution sectors per range.

The caller passes an array that specifies a series of ranges. For each range specified in the array, the intrinsic returns the total free space for free areas found in that range.

Refer to Items 36 and 37 for the format of the free space distribution array passed to the intrinsic and for the format of the returned array.

40 &amp; 41

### **Total free space**

*Function* = MPE/iX: DISCFREE , MPE V: FREE5

Returns the total free space on a volume or a group of volumes.

Item 40 returns a 64-bit signed integer containing the total free space in sectors.

Item 41 returns a 64-bit real.

When the specifier is 1 or 4, the total free space consists of the total free space on the volume whose LDEV or volume name was specified.

## HPVOLINFO

When the specifier is 2 or 3, the total free space on the volumes that make up the volume set or class is returned.

42 & 43

### **Largest contiguous free space area**

*Function* = MPE/iX: DISCFREE , MPE V: FREE5

Returns the largest contiguous free space area on a volume or a group of volumes.

Item 42 returns a 64-bit signed integer containing the largest contiguous free space area in sectors.

Item 43 returns a 64-bit real.

When the specifier is 1 or 4, the largest contiguous free space consists of the largest contiguous free space on the volume whose LDEV or volume name was specified.

When the specifier is 2 or 3, the largest contiguous free space on the volumes that make up the volume set or class is returned.

For example, for specifiers 2 or 3, if a volume set is composed of LDEVs 2, 3, and 4, and the largest contiguous free space on these volumes are 37785, 56528, and 56171, respectively, the value returned is 56528.

## Variable Conventions

Convention	Meaning
I16	16-bit signed integer
I32	32-bit signed integer
I64	64-bit signed integer
R64	64-bit signed real
CA	Character array
I64A	64-bit signed integer array
R64A	64-bit signed real array

ITEM #	ITEM	TYPE	APPLICABLE VOLUME SPECIFIERS
2	Number of volume sets	I32	0
3	List of volume set names	CA	0
4	Number of volume classnames	I32	1, 2, 4
5	List of volume class names	CA	1, 2, 4
6	Number of member volumes	I32	2, 3, 5
7	List of member volume names	CA	2, 3, 5
8	Drive type	CA	1, 4
9	Sector size in bytes	I32	1, 4
10	Volume type	I32	1, 4
11	Volume name	CA	1
12	Volume set name	CA	1
13	Logical device number	I16	4
14	Volume capacity in sectors	I64	1, 2, 3, 4
15	Volume capacity in sectors	R64	1, 2, 3, 4
16	Total MPE overhead in sectors	I64	1, 2, 3, 4
17	Total MPE overhead in sectors	R64	1, 2, 3, 4
18	MPE/iX transient space overhead in sectors	I64	1, 2, 3, 4

**HPVOLINFO**

<b>ITEM #</b>	<b>ITEM</b>	<b>TYPE</b>	<b>APPLICABLE VOLUME SPECIFIERS</b>
19	MPE/iX transient space overhead in sectors	R64	1, 2, 3, 4
20	Configured maximum MPE/iX transient space in sectors	I64	1, 2, 3, 4
21	Configured maximum MPE/iX transient space in sectors	R64	1, 2, 3, 4
22	MPE V virtual memory overhead in sectors	I64	1, 2, 4
23	MPE V virtual memory overhead in sectors	R64	1, 2, 4
24	Directory space overhead in sectors	I64	1, 2, 3, 4
25	Directory space overhead in sectors	R64	1, 2, 3, 4
26	MPE/iX file label overhead in sectors	I64	1, 2, 3, 4
27	MPE/iX file label overhead in sectors	R64	1, 2, 3, 4
28	MPE/iX transaction management overhead	I64	1, 2, 3, 4
29	MPE/iX transaction management overhead	R64	1, 2, 3, 4
30	Spoolfile disk space usage in sectors	I64	1, 2, 3, 4
31	Spoolfile disk space usage in sectors	R64	1, 2, 3, 4
32	Disk space used by permanent files in sectors	I64	1, 2, 3, 4
33	Disk space used by permanent files in sectors	R64	1, 2, 3, 4
34	(Reserved for) Disk space used by temporary files in sectors	I64	1, 2, 3, 4
35	(Reserved for) Disk space used by temporary files in sectors	R64	1, 2, 3, 4
36	Free space distribution array	I64A	1, 2, 3, 4
37	Free space distribution array	R64A	1, 2, 3, 4
38	Free space distribution sectors per range	I64A	1, 2, 3, 4

<b>ITEM #</b>	<b>ITEM</b>	<b>TYPE</b>	<b>APPLICABLE VOLUME SPECIFIERS</b>
39	Free space distribution sectors per range	R64A	1, 2, 3, 4
40	Total free space in sectors	I64	1, 2, 3, 4
41	Total free space in sectors	R64	1, 2, 3, 4
42	Largest contiguous free space area in sectors	I64	1, 2, 3, 4
43	Largest contiguous free space area in sectors	R64	1, 2, 3, 4

---

## INITUSLF

NM and CM callable.

Initializes a buffer corresponding to record 0 of a USL file to the empty state.

---

**Note** A USL contains CM object code and is meaningful only in the CM program development process.

---

### Syntax

I16	I16V	I16A
<i>uslferror</i> := INITUSLF( <i>uslfnum</i> , <i>record</i> );		

### Functional Return

*uslferror*     **16-bit signed integer (assigned functional return)**

Returns error information. If no error occurs, no value is returned. If an error occurs (condition code returns CCL (1)), one of the following is returned:

Value	Meaning
0	An unexpected end-of-file was encountered when writing to <i>uslfnum</i> .
1	Unexpected I/O error occurred.
3	Your request attempted to exceed the maximum file size (32,768 records) or was smaller than the minimum file size (4 records).

### Parameters

*uslfnum*     **16-bit signed integer by value (required)**

The file number of the USL file.

*record*     **16-bit signed integer array (required)**

Passes and returns a buffer, corresponding to the first record of the USL file (record 0), to be initialized to the empty state. Set this 128-element array to all zeros. The intrinsic sets certain values in record 0 before returning to the calling program.



**Condition Codes**

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. An error number is returned to <i>uslerror</i> .

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>MPE Segmenter Reference Manual</i> (30000-90011)

---

## IODONTWAIT

NM and CM callable.

Initiates completion operations for an I/O request.

### Syntax

I16	I16V	UDS	I16	U16
<i>fnum:=IODONTWAIT(filenum,buffer,length,cstation)</i>				

### Functional Return

*fnum*            **16-bit signed integer (assigned functional return)**

The file number for which the completion occurred. If no completion occurred, zero is returned.

### Parameters

*filenum*        **16-bit signed integer by value (required)**

The file number for which there is a pending I/O request. If you specify 0, the IODONTWAIT intrinsic checks for any I/O completion.

*buffer*         **user-defined structure (optional)**

Passes an input buffer. This buffer must be large enough to contain the input record. It should be the same buffer specified in the original I/O request if that was a read request. This allows for a proper recognition of EOF where applicable. The buffer parameter is required if not in privileged mode, or the buffer parameter can be omitted if the GETPRIVMODE intrinsic is called first.

*length*         **16-bit signed integer by reference (optional)**

Returns the length of the received or transmitted record. If the original request specified a byte count, the integer represents bytes; if the request specified half words, the integer represents half words. This parameter is pertinent only if the original I/O request was a read request. The FREAD intrinsic always returns zero as its functional return if you specify nowait I/O. In this case, the actual record length is returned in the *length* parameter of IODONTWAIT.

*cstation*       **16-bit unsigned integer by reference (optional)**

Returns the number of the calling station when completed (used for distributed systems).

(ASC) Not used for asynchronous devices.

## Operation Notes

The IODONTWAIT intrinsic operates the same as IOWAIT with one exception: if you call IOWAIT and no I/O has completed, then the calling process is suspended until some I/O completes; if you call IODONTWAIT and no I/O has completed, then control is returned to the calling process. (CCE (2) is returned and the result of IODONTWAIT is zero.)

## Condition Codes

- CCE (2) Request granted. If the functional return is not zero, then I/O completion occurred with no errors. If the functional return is zero, no I/O has completed.
- CCG (0) An end-of-file was encountered.
- CCL (1) Request denied. Normal I/O completion did not occur; there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred.

## Related Information

- Intrinsics None
- Commands None
- Manuals *Interprocess Communication Programmer's Guide* (32650-90019)

---

## IOWAIT

NM and CM callable.

Initiates completion operations for an I/O request.

### Syntax

I16	I16V	UDS	I16	U16
<i>fnum</i> := IOWAIT( <i>filenum</i> , <i>buffer</i> , <i>length</i> , <i>cstation</i> );				

### Functional Return

*fnum*            **16-bit signed integer (assigned functional return)**

The file number for which the completion occurred. If no completion occurred, zero is returned.

### Parameters

*filenum*        **16-bit signed integer by value (required)**

The file number for which there is a pending I/O request. If you specify 0, the IOWAIT intrinsic waits for the first I/O completion.

*buffer*         **user-defined structure (optional)**

Passes an input buffer. This buffer must be large enough to contain the input record. It should be the same buffer specified in the original I/O request if that was a read request. This allows for proper recognition of EOF where applicable. The buffer parameter is required if not in privileged mode, or the buffer parameter can be omitted if the GETPRIVMODE intrinsic is called first.

*length*         **16-bit signed integer by reference (optional)**

Returns the length of the received or transmitted record. If the original request specified a byte count, the integer represents bytes; if the request specified half words, the integer represents half words. This parameter is pertinent only if the original I/O request was a read request. The FREAD intrinsic always returns zero as its functional return if you specify nowait I/O. In this case, the actual record length is returned in the *length* parameter of IOWAIT.

*cstation*        **16-bit unsigned integer by reference (optional)**

Returns the number of the calling station when completed (used for distributed systems).

(ASC) Not used for asynchronous devices.

## Operation Notes

The IOWAIT intrinsic initiates completion operations for an I/O request. If you opened a file with the nowait I/O mode *aoption* of the FOPEN/HPFOPEN intrinsic (*aoption* bit (4:1) = 1), calls to either the IOWAIT or IODONTWAIT intrinsic must follow all read and write requests. This intrinsic initiates completion operations for the associated I/O request, including data transfer into your buffer area if necessary.

The IOWAIT intrinsic call must precede any subsequent I/O request against the file. Within this restriction, you can delay the IOWAIT intrinsic call as long as desired to allow effective I/O and processing overlap.

## Condition Codes

- |         |  |
|---------|--|
| CCE (2) | Request granted. I/O completion occurred with no errors.   |
| CCG (0) | An end-of-file was encountered.  |
| CCL (1) | Request denied. Normal I/O completion did not occur, because there were no I/O requests pending, a parameter error occurred, or an abnormal I/O completion occurred. |

## Related Information

- |            |  |
|------------|--|
| Intrinsics | None   |
| Commands   | None   |
| Manuals    | <i>Interprocess Communication Programmer's Guide</i> (32650-90019) |

---

## JOBINFO

NM and CM callable.

Returns information about any current job or session under your user name and account. If you have sufficient capabilities, it returns information about jobs or sessions running under other users and accounts.

### Syntax

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

---

**Note** You can specify up to five *itemnum/item/itemerror* triples.

---

### Parameters

*jsind*      **16-bit signed integer by value (required)**  
Indicates whether *jsnum* denotes a session or job:

Value	Meaning
1	<i>jsnum</i> is a session
2	<i>jsnum</i> is a job

*jsnum*      **32-bit signed integer by reference (required)**  
Passes the job/session number for which information is requested. Job/session numbers can be displayed by the **SHOWME** and **SHOWJOB** commands. If this value is zero, **JOBINFO** returns information about your logon job/session (unless the first *itemnum* has a value of 1; see below). When you pass a 0 for *jsnum*, **JOBINFO** returns the actual job/session number in *jsnum*.

***jsstatus* 16-bit unsigned integer array (required)**

Returns indicator of the success or failure of the call in a two-element array. The first element contains the error indicator. The second element should be ignored.

Value	Meaning
0	Successful call. All <i>itemerrors</i> equal zero.
1	Partially successful call. One or more <i>itemerror(s)</i> returned with nonzero values.
2	Unsuccessful call. All <i>itemerrors</i> returned with nonzero values.
3	Unsuccessful call. Syntax error in calling sequence.
4	Unsuccessful call. Unable to retrieve <i>jsnum</i> .
5	Process terminated. The process terminated during the start of retrieval.

***itemnum* 16-bit signed integer by value (optional)**

Specifies which item value is to be returned. Refer to Table 4-20 for a complete list of valid item numbers and what they return to the caller.

***item* type varies (optional)**

A reference parameter (whose data type corresponds to the data type for the desired information) to which the desired information is returned.

These are standard data types and are defined in Table 2-1. All *itemnum* and *item* parameters are output parameters. The only exception is *itemnum=1*, when all the following conditions are met:

- The caller passes *itemnum=1* in the first item triple.
- The *item* parameter contains a character string of the form *[jsname,]username.acctname*.
- The *jsnum* parameter is set to 0, and the *jsind* parameter is set to 1 or 2.

If all these conditions are met, JOBINFO returns information about the specified job/session in all other items and returns that job/session number in *jsnum*. The size of the first item is determined by the way it is used.

## JOBINFO

*itemerror*

### 16-bit signed integer by reference (optional)

Returns the success or failure of the retrieval of each item. The returned values are:

Value	Meaning
0	Successful
1	Invalid <i>itemnum</i>
2	Information requested not valid for specified <i>jsnum</i>
3	Insufficient capability
4	Information not available

**Table 4-22. JOBINFO Itemnum/Item Values**

Itemnum	Mnemonic	Item Description
1	CA	[jsname,]user.account. Can be used as an I/O parameter: <ul style="list-style-type: none"><li>■ If used as an input parameter, the array must be 26 bytes long, plus one for a binary 0 (zero) terminator. The input string must be in the form [jsname,]user.account. The wildcard character @ is not allowed.</li><li>■ If used as an output parameter, the array must be 26 bytes long. The output is left-justified and padded with blanks.</li></ul>
2	CA8	Session/job name. An ASCII output parameter. Output is left-justified and padded with blanks.
3	CA8	User name. An ASCII output parameter. Output is left-justified and padded with blanks.
4	CA8	User logon group. An ASCII output parameter. Output is left-justified and padded with blanks.
5	CA8	User account. An ASCII output parameter. Output is left-justified and padded with blanks.
6	CA8	User home group. An ASCII output parameter. Output is left-justified and padded with blanks.
7	I32	The date/time that is displayed by the :SHOWJOB command. This date/time is updated whenever a job changes state (SCHED, WAIT, EXEC). Returns a 32-bit signed integer to be used by the FMTCLOCK intrinsic. For a job, this is the time it enters the WAIT state.



Table 4-22. JOBINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
8	U16	The date/time that is displayed by the <code>:SHOWJOB</code> command. This date/time is updated whenever a job changes state (SCHED, WAIT, EXEC).  Returns a 16-bit unsigned integer to be used by the <code>FMTCALENDAR</code> intrinsic. For a job, this is the date when it enters the <code>WAIT</code> state.
9	CA8	Input LDEV/class name.  An ASCII output parameter. Whatever array the user passes in will be overwritten with the LDEV/class name, using exactly the number of characters in the LDEV/class name. The remainder of the array is left intact; no fill character is used.
10	CA8	Output LDEV/class name.  An ASCII output parameter. Whatever array the user passes in will be overwritten with the LDEV/class name, using exactly the number of characters in the LDEV/class name. The remainder of the array is left intact; no fill character is used.
11	CA	Current job step.  Returns a maximum of 238 ASCII characters and is the image of the command currently executing. The array must be long enough for the expected command image (the length can be obtained from <code>itemnum=34</code> ).
12	I16	Current number of active jobs
13	I16	Current number of active sessions
14	I16	Job input priority
15	I32	Job/session number
16	I16	Jobfence
17	I16	Job output priority
18	I16	Number of copies
19	I16	Job limit (system)
20	I16	Session limit (system)
21	I16	Job deferred. Returns the values:  0 No 1 Yes
22	U16	Main CI PIN for job/session
23	U16	Original job spooled.  Returns the values:  0 No 1 Yes

**JOBINFO**

**Table 4-22. JOBINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
24	U16	RESTART option. Returns the values: 0 No 1 Yes
25	U16	Sequenced job. Returns the values: 0 No 1 Yes
26	U16	Term code. Returns the values: 0 Regular terminal 1 Regular terminal with special logon
27	U16	CPU limit
28	U16	Session/job state. Returns the values: 2 Executing 4 Suspending 32 Wait 48 Initialization 56 Scheduled
29	U32	User's local attributes
30	I16	\$STDIN spoolfile number. Returns data for current jobs and sessions only. Valid for spoolfile numbers less than or equal to 32768.
31	I16	\$STDIN spoolfile status. Returns data for current jobs and sessions only. Returns the values: 0 Open 1 Active 3 Reserved 4 Ready
32	I16	\$STDLIST spoolfile number. Returns data for current jobs and sessions only. Valid for spoolfile numbers less than or equal to 32768.

Table 4-22. JOBINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
33	I16	<p>\$STDLIST spoolfile status.</p> <p>Returns data for current jobs and sessions only. Returns the values:</p> <p>0 Active 1 Ready 2 Open 3 Reserved</p>
34	I16	Length of <i>itemnum</i> =11 (current job step)
35	U16	<p>SET STDLIST=DELETE invoked. Returns the values:</p> <p>0 \$STDLIST is saved 1 SET STDLIST=DELETE is invoked</p>
36	U16	Job information table data segment number
37	N/A	Not allowed. Call results in error to calling routine
38	I32	<p>\$STDIN spoolfile number.</p> <p>Returns data for current jobs and sessions only. Valid for all spoolfile numbers regardless of size.</p>
39	I32	<p>\$STDLIST spoolfile number.</p> <p>Returns data for current jobs and sessions only. Valid for all spoolfile numbers regardless of size.</p>
40	U16	<p>Session quiet mode.</p> <p>Returns:</p> <p>0 No 1 Yes</p>
41	I32	The job or session number of the job submitter.
42	CA8	<p>The job or session name of the job submitter.</p> <p>An ASCII output parameter. Output is left justified and padded with blanks.</p>
43	CA8	<p>The user name of the job submitter.</p> <p>An ASCII output parameter. Output is left justified and padded with blanks.</p>
44	CA8	<p>The user account of the job submitter.</p> <p>An ASCII output parameter. Output is left justified and padded with blanks.</p>
45	CA8	<p>The logical device of the job submitter.</p> <p>An ASCII output parameter. Output is left justified and padded with blanks.</p>

## JOBINFO

**Table 4-22. JOBINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
46	U16	<p>The date/time that the <b>STREAM</b> command was submitted by the job submitter. For jobs, this is the date that the job was streamed (and is the same date that appears in the job submitter banner.) For sessions, this is the date that the session was started via either the <b>STARTSESS</b> command, the <b>STARTSESS</b> intrinsic, or by the default automatic logon at boot time.</p> <p>This item value differs from item 8 only for jobs scheduled for the future.</p> <p>Returns a 16-bit unsigned integer to be used by the <b>FMTCALENDAR</b> intrinsic.</p>
47	I32	<p>The date/time that the <b>STREAM</b> command was submitted by the job submitter. For jobs, this is the time that the job was streamed (and is the same time that appears in the job submitter banner.) For sessions, this is the date that the session was started via either the <b>STARTSESS</b> command, the <b>STARTSESS</b> intrinsic, or by the default automatic logon at boot time.</p> <p>This item value differs from item 7 only for jobs scheduled for the future.</p> <p>Returns a 32-bit signed integer to be used by the <b>FMTCLOCK</b> intrinsic.</p>
48	U16	<p>The job or session type of the job submitter.</p>

---

**Note** Unless otherwise indicated, all mnemonics are as defined in Table 2-1.

---

### Operation Notes

System manager (SM) or account manager (AM) capability is required to retrieve information about jobs and sessions not logged under your user and account. AM capability restricts access to account session and job information; SM capability allows access to all session and job information. This intrinsic does not alter the condition code.

### Related Information

Intrinsics      **JOBINFO**

Commands      **SHOWJOB, SHOWME, SHOWOUT, SHOWIN**

Manuals      *Getting System Information Programmer's Guide (32650-90018)*

---

## KILL

NM and CM callable.

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

### Syntax

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

### Parameters

*pin*            **16-bit signed integer by value (required)**  
Passes the process identification number (PIN) of the child process to be deleted.

### Operation Notes

All resources held by the deleted processes are released. Any remaining files opened by the deleted processes are closed and assigned the same disposition they had when opened.

### Condition Codes

CCE (2)        Request granted.  
CCG (0)        Request granted. The specified process is already being terminated.  
CCL (1)        Request denied. An illegal PIN was specified.

### Related Information

Intrinsics     None  
Commands      None  
Manuals        *Process Management Programmer's Guide* (32650-90023)

---

## LOADPROC

NM and CM callable.

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

### Syntax

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

### Functional Return

*idnum*      **16-bit signed integer (assigned functional return)**

Returns an identity number required by the UNLOADPROC intrinsic to unload the CM SL procedure dynamically. If a loader error occurs (condition code returns CCL (1)), the identity number represents a CM loader error number.

### Parameters

*procname*    **character array (required)**

Passes the name of the procedure to be loaded. The procedure name must be terminated with a blank and be <= 15-characters; excess characters are truncated without a warning message and any procedure matching the truncated name is loaded.

*library*     **16-bit signed integer by value (required)**

Passes a value requesting a library search for the procedure residing in your logon group:

Value	Meaning
0	Search system SL only
1	Search logon account SL, then system SL
2	Search logon group SL, logon account SL, then system SL
3	Search program file's account SL, then system SL
4	Search program file's group SL, program file's account SL, and then system SL

*plabel*      **16-bit signed integer by reference (required)**

Returns the procedure's label. This is the external CM *plabel*.

**Condition Codes**

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. The value returned to <i>idnum</i> is a CM loader error code.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Switch Programming Guide</i> (32650-90014)

---

## 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.

### Syntax

I16V      U16      CA
LOCKGLORIN( <i>rinum</i> , <i>lockflag</i> , <i>rinpassword</i> );

### Parameters

*rinum*      **16-bit signed integer by value (required)**

Passes the RIN of the resource to be locked. This is the RIN furnished in the GETRIN command.

*lockflag*      **16-bit unsigned integer by reference (required)**

Passes and returns conditional or unconditional RIN locking specification through bit (15:1):

Value	Meaning
0	Locking takes place only if the RIN is currently unlocked. If the RIN is not unlocked, control returns to the calling process immediately with the condition code CCG (0).
1	Locking takes place unconditionally. If the RIN is currently locked, the calling process suspends until the RIN becomes unlocked.

All other bits are ignored. See the discussion of condition codes for the values returned in *lockflag*.

*rinpassword*      **character array (required)**

Passes the RIN password assigned through the GETRIN command. This array must be a minimum of 10 bytes in length and must be terminated by a nonalphanumeric ASCII character; a blank is recommended.

### Condition Codes

The following condition codes are possible if *lockflag* bit (15:1)=1:

- CCE (2)      Request granted. If the calling process has already locked the RIN, *lockflag* bit (15:1) is set to 0. If the RIN is free, *lockflag* bit (15:1) is set to 1.
- CCG (0)      Not returned.
- CCL (1)      Request denied. Invalid RIN was specified. Either *rinum* is not a global RIN, or the value is out of bounds for the RIN table.



The following condition codes are possible if *lockflag* bit (15:1)=0:

- CCE (2) Request granted. If the calling process has already locked the RIN, *lockflag* bit (15:1) is set to 0. If the RIN is free, *lockflag* bit (15:1) is set to 1.
- CCG (0) Request denied. The RIN was locked by another job.
- CCL (1) Request denied because of invalid RIN. Either *rinum* is not a global RIN, or the value is out of bounds for the RIN table.

## Operation Notes

The LOCKGLORIN intrinsic can lock any global RIN assigned to a group of users, one process at a time. When this is done, any other processes that attempt to lock this RIN are suspended.

To use the LOCKGLORIN intrinsic, the RIN number and the RIN password must be known. Multiple RIN (MR) capability is required to lock two or more global RINs simultaneously. If you do not have MR capability and attempt to lock two or more RINs simultaneously, the LOCKGLORIN intrinsic aborts the calling process and prints the following error message:

```
ONLY ONE RIN CAN BE LOCKED
```

Lock global RINs using the LOCKGLORIN and FLOCK intrinsics.

## Related Information

- Intrinsics None
- Commands None
- Manuals *Resource Management Programmer's Guide* (32650-90024) and *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364)

---

## LOCKLOCRIN

NM and CM callable.

Locks a local resource identification number (RIN).

### Syntax

I16V    U16 LOCKLOCRIN( <i>rinum</i> , <i>lockflag</i> );
--

### Parameters

*rinum*    **16-bit signed integer by value (required)**

A previously allocated local RIN; valid values range from 1 to the value specified in the *rincount* parameter of the GETLOCRIN intrinsic.

*lockflag*    **16-bit unsigned integer by reference (required)**

Passes and returns conditional or unconditional RIN locking specification through bit (15:1).

Value	Meaning
0	Locking takes place only if the RIN is currently unlocked. If it is not, control returns to the calling process immediately with the condition code CCG (0).
1	Locking takes place unconditionally. If the RIN is currently locked, the calling process suspends until the RIN becomes unlocked.

All other bits are ignored. See the discussion of condition codes for the values returned in *lockflag*.

### Operation Notes

Any local RIN assigned to a job can be locked, one process at a time, by using the LOCKLOCRIN intrinsic. When this is done, other processes within the job that attempt to lock that RIN are suspended until the locked RIN is released.

## Condition Codes

The following condition codes are possible if *lockflag* bit (15:1)=1:

- CCE (2) Request granted. If the calling process has already locked the RIN, *lockflag* bit (15:1) is set to 1. If the RIN is free, *lockflag* bit (15:1) is set to 0.
- CCG (0) Not returned.
- CCL (1) Request denied. The RIN is invalid; the *rinum* was too large, no local RIN was allocated, or *rinum* specified a number less than or equal to zero.

The following condition codes are possible if *lockflag* bit (15:1)=0:

- CCE (2) Request granted. If the calling process has already locked the RIN, *lockflag* bit (15:1) is set to 1. If the RIN is free, *lockflag* bit (15:1) is set to 0.
- CCG (0) Request denied. The RIN was locked by another process.
- CCL (1) Request denied. The RIN is invalid; the *rinum* was too large, no local RIN was allocated, or *rinum* specified a number less than or equal to zero.

## Related Information

- Intrinsics None
- Commands None
- Manuals *Resource Management Programmer's Guide* (32650-90024).

---

## LOCRINOWNER

NM and CM callable.

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

### Syntax

I16	I16V
<i>pin</i> :=LOCRINOWNER( <i>rinum</i> );	

### Functional Return

*pin*            **16-bit signed integer (assigned functional return)**

If the RIN is locked by the parent of the calling process, **LOCRINOWNER** returns 0. If the RIN is locked by any other process in your process structure, **LOCRINOWNER** returns the PIN of that process.

### Parameters

*rinum*            **16-bit signed integer by value (required)**

Passes the number of the local RIN for which the PIN of the locking process is to be determined. The range of valid values is from 1 to the value specified in the *rincount* parameter of the **GETLOCRIN** intrinsic.

### Operation Notes

The **LOCRINOWNER** intrinsic determines the PIN of the process that has a particular local RIN locked. After a process has acquired local RINs, other processes in the process structure can lock and unlock these RINs.

### Condition Codes

- CCE (2)            Request granted.
- CCG (0)            Request denied. The local RIN specified by *rinum* is not currently locked by any process.
- CCL (1)            Request denied. The *rinum* parameter was invalid (that is, *rinum* was less than or equal to 0, greater than the RIN table size, or greater than the number of local RINs currently allocated to this process structure).

**Related Information**

Intrinsics       None

Commands       None

Manuals        *Resource Management Programmer's Guide* (32650-90024).

---

## 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.

### Syntax

I32V    I16                    I16V    *
LOGINFO ( <i>index,logstatus</i> [, <i>itemnum,item</i> ] [...]);

---

**Note**            You can specify up to four *itemnum/item* pairs.

---

### Parameters

*index*            **32-bit signed integer by value (required)**

Passes your access to the user logging system. This is the value returned in the *index* parameter of the OPENLOG intrinsic.

*logstatus*        **16-bit signed integer by reference (required)**

Returns one of the following values, indicating the success/failure of the intrinsic call:

Value	Meaning
0	Successful
2	Parameter out of bounds
4	Incorrect <i>index</i> parameter
7	Illegal capability; user logging (LG) or system supervisor (OP) capability required
14	Invalid access
17	Either <i>itemnum</i> or <i>item</i> missing
18	Invalid <i>itemnum</i>

*itemnum*          **16-bit signed integer by value (optional)**

Specifies which *item* value is to be returned. (Refer to Table 4-23)

*item*              **array whose type varies (optional)**

Returns the value of the item specified by the corresponding *itemnum*. Refer to Table 4-23.

Table 4-23. LOGINFO Itemnum/Item Values

Itemnum	Mnemonic	Item Description
1	I32	Total number of records written in current file
2	I32	Current file size
3	I32	Current file space left
4	I16	Number of users
5	I32	Total records written in whole file set
6	CA	Current log file name
7	I16	Current log file type: 0 DISK 1 TAPE
8	CA	Previous log file name
9	I16	Previous log file type: 0 DISK 1 TAPE
10	U16	CHANGELOG allowed (true/false)
11	U16	AUTO allowed (true/false)
12	I16	Current file sequence number
13	I16	Log status: 0 INACTIVE 1 ACTIVE 2 Close pending 3 Stop pending

### Related Information

Intrinsics      None

Commands      SHOWLOGSTATUS

Manuals      *User Logging Programmer's Guide (32650-90027)*

---

## LOGSTATUS

NM and CM callable.

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

### Syntax

I32    U16A    I16 LOGSTATUS( <i>index</i> , <i>loginfo</i> , <i>logstatus</i> );
--

### Parameters

- index*            **32-bit signed integer by reference (required)**
- Passes your access to the user logging system. This is the value returned in the *index* parameter of the OPENLOG intrinsic.
- loginfo*           **16-bit unsigned integer array (required)**
- Returns the following information:
- The total records written in current logging file (a 32-bit value in elements 0 and 1 of *loginfo*).
  - The size, in records, of the current logging file (a 32-bit value in elements 2 and 3 of *loginfo*).
  - The space, in records, remaining in the current logging file (a 32-bit value in elements 4 and 5 of *loginfo*).
  - The number of users using the logging file (a 16-bit value in element 6 of *loginfo*).
- logstatus*        **16-bit signed integer by reference (required)**
- Returns one of the following values, indicating the success/failure of the intrinsic call:

Value	Meaning
0	Successful
2	Parameter out of bounds
4	Incorrect <i>index</i> parameter
7	Illegal capability; user logging (LG) or system supervisor (OP) capability required
14	Invalid access



**Related Information**

Intrinsics	None
Commands	SHOWLOGSTATUS
Manuals	<i>User Logging Programmer's Guide (32650-90027)</i>

---

## MAIL

NM and CM callable.

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

### Syntax

U16	I16V	I16
<i>mailstatus</i> :=MAIL( <i>pin</i> , <i>length</i> );		

### Functional Return

*mailstatus*      **16-bit unsigned integer (assigned functional return)**

Returns the status of the mailbox:

Value	Meaning
0	Mailbox empty.
1	Contains previous outgoing mail from this calling process that the destination process has not yet collected.
2	Contains incoming mail awaiting collection by this calling process. The length of the mail is returned in <i>length</i> .
3	An error occurred, an invalid <i>pin</i> was specified, or a bounds check failed.
4	Temporarily inaccessible, other intrinsics are using it in the preparation or analysis of mail.

### Parameters

*pin*      **16-bit signed integer by value (required)**

Passes the mailbox to be tested. If *pin* specifies the mailbox of a child process, it must be the process identification number (PIN) of that child. Zero specifies the mailbox of a parent process.

*length*      **16 bit signed integer by reference (required)**

Returns the length, in half words, of any incoming mail in the mailbox.

## Operation Notes

If the mailbox contains mail that is awaiting collection by this process, the length of this message (in half words) is returned to the calling process in the *length* parameter. This enables the calling process to initialize its stack in preparation for receipt of the message.

## Condition Codes

- CCE (2) Request granted. The mailbox *mailstatus* was tested.
- CCG (0) Request denied. An illegal *pin* parameter was specified. The value of 3 is returned to the calling process through *mailstatus*.
- CCL (1) Request denied. The *length* parameter defines stack address that does not have write access.

## Related Information

- Intrinsics None
- Commands None
- Manuals *Interprocess Communication Programmer's Guide* (32650-90019)

---

## MERGEEND

NM and CM callable.

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

### Syntax

```
MERGEEND;
```

### Condition Codes

CCE (2) Request granted. No error occurred during the **MERGE** program.

CCG (0) Not returned.

CCL (1) Request denied. An error occurred during the **MERGE** program.

### Operation Notes

Call this intrinsic only if the **MERGEINIT** intrinsic was called.

### Related Information

Intrinsics HPSORT, HPMERGE

Commands None

Manuals *SORT-MERGE/XL Programmer's Guide* (32650-90080)

---

## 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.

### Syntax

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

### Parameters

<i>errorcode</i>	<b>16-bit signed integer by value</b> Returns the error number from the MERGEINIT <i>errorparm</i> parameter.
<i>message</i>	<b>character array</b> Returns the text of the error message. This parameter must be at least 80 characters long.
<i>length</i>	<b>16-bit signed integer</b> Returns the length of the <i>message</i> parameter.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## MERGEINIT

NM and CM callable.

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

### Syntax

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

### Parameters

- inputfiles*     **16-bit signed integer array (optional)**  
Passes the file identification numbers of the input files to be merged. The last element of this array should be set to zero. If the files are opened with either the **NOBUF** or **MR** (multirecord) access option (*aoption*), the MERGE program performs the buffering and blocking/deblocking. \$NULL is not a valid input file.
- preprocessor*   **procedure (optional)**  
Do not specify, maintain parameter position.
- outputfiles*    **16-bit signed integer array (optional)**  
Passes the file identification number of the output file. The second array element must be a zero, indicating the end of the array. If the files are opened with either the **NOBUF** or **MR** (multi-record) access option (*aoption*), the MERGE program performs the buffering and blocking/deblocking.
- postprocessor*   **procedure (optional)**  
Do not specify, maintain parameter position.
- length*        **16-bit signed integer**  
Denotes the number of characters in the record.

*keysonly*      **16-bit signed integer by value (optional)**

Passes a flag that determines if only keys are sent as output. If true, the key fields are concatenated together with the major key on the left and the remaining keys following. If false, the entire record is sent as output.

Default: false.

*numkeys*      **16-bit signed integer (optional)**

Passes the number of keys used during the comparison of records. This parameter can be either equal to or greater than one. If you specify the *numkeys* parameter, you must also specify the *keys* parameter. Together, *numkeys* and *keys* describe the way records are merged.

*keys*      **16-bit signed integer array (optional)**

Passes information about the keys used during comparison of records. If you specify the *keys* parameter, you must specify the *numkeys* parameter. Together, *keys* and *numkeys* describe the way records are merged.

The first element gives the position of the first character of the key within the record. The second element gives the number of characters in the key. Bits (0:8) of the third element give the ordering sequence of the records (0 for ascending, 1 for descending). Bits (8:8) of the third element indicate the type of data according to the following convention:

Value	Meaning
0	8-Bit unsigned integer
1	Twos complement (including 16-bit and 32-bit signed integer)
2	Floating-point (including real and long)
3	Packed decimal
4	Display trailing sign
5	Packed decimal with even number of digits
6	Display leading sign

## MERGEINIT

Value	Meaning
7	Display leading sign separate
8	Display trailing sign separate
9	Character (using the collating sequence of <i>charseq</i> )
11	Short floating-point decimal
12	Floating-point decimal

---

**Note** The integrity of the *keys* array must be maintained throughout the MERGE operation. Do not change it until after you have called the **MERGEEND** intrinsic.

---

*altseq*      **16-bit signed integer array (optional)**

Passes an alternate collating sequence. Bits (0:8) of the first element are specified according to the following table, where the sequence comprises the columns and the data comprises the rows:

	ASCII	EBCDIC	ALTSEQ
ASCII	chr(255)	chr(2)	chr(0)
EBCDIC	char(1)	chr(255)	undefined

Bits (8:8) of the first element specify one fewer than the total number of characters in the collating sequence (in this instance, chr(255) or %377).

The remaining array elements comprise the actual collating sequence responsible for the particular MERGE operation.

*keycompare*      **procedure (reserved)**

Do not specify this parameter, but maintain parameter position.

*errorproc*      **procedure (reserved)**

Do not specify this parameter, but maintain parameter position.



*statistics*      **16-bit signed integer array (optional)**

Returns information on the MERGE operation, as follows:

Value	Meaning
1	Number of input files
2,3	Number of merged records (32-bit integer)
4	Space available for merging
5,6	Number of comparisons (32-bit integer)
7,8	CPU time in milliseconds (32-bit integer)
9,10	Elapsed time in milliseconds (32-bit integer)

*failure*      **16-bit signed integer (optional)**

Returns a true value (-1) if a fatal error occurs; otherwise *failure* returns a false value (0). Refer also to the discussion of condition codes.

*errorparm*      **16-bit signed integer (optional)**

Returns the error number if an error occurs. Use the MERGEERRORMESS intrinsic to obtain the error message text.

*spaceallocation*      **16-bit signed integer (reserved)**

Do not specify this parameter, but maintain parameter position.

*charseq*      **16-bit signed integer array (optional)**

Passes language information in a two-element array. Set the first element to 1. Set the second element to the language ID number of the native language whose collating sequence is to be used to sort keys of type 9 (character).

## MERGEINIT

### Condition Codes

- CCE (2) Request granted. No error occurred during the MERGE program. The value of the *failure* parameter of the MERGEINIT intrinsic is set to false.
- CCG (0) Not returned.
- CCL (1) Request denied. An error occurred during the MERGE program. The value of the *failure* parameter of the MERGEINIT intrinsic is set to true.

### Related Information

- Intrinsics HPSORT, HPMERGE
- Commands None
- Manuals *SORT-MERGE/XL Programmer's Guide* (32650-90080)

---

## MERGEOUTPUT

NM and CM callable.

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

### Syntax

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

### Parameters

<i>record</i>	<b>16-bit character array</b> Returns the next output record.
<i>length</i>	<b>16-bit signed integer</b> Returns the length of the <i>record</i> parameter value.

### Operation Notes

MERGEOUTPUT requests records from MERGEINIT, one at a time, if the *outputfiles* parameter in the call to the MERGEINIT intrinsic was not specified. Call MERGEOUTPUT from the program after a call to MERGEINIT, but before a call to MERGEEND.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080).

---

## MERGESTAT

NM and CM callable.

Prints the MERGE program statistics on \$STDLIST.

### Syntax

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

### Parameters

*statistics*      **16-bit signed integer array**

Returns MERGE program statistics, such as the number of input files, the number of records, the space available, the number of compares, the CPU time in minutes, and the elapsed time in minutes.

### Operation Notes

Call MERGESTAT after you have called the MERGEEND intrinsic.

### Related Information

Intrinsics      HPSORT, HPMERGE

Commands      None

Manuals      *SORT-MERGE/XL Programmer's Guide* (32650-90080).

## 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.

### Syntax

```
MERGETITLE;
```

### Operation Notes

MERGETITLE can be called from a program any time after the system intrinsics have been declared.

### Related Information

Intrinsics      HPSORT, HPMERGE

Commands        None

Manuals         *SORT-MERGE/XL Programmer's Guide* (32650-90080).

---

## MYCOMMAND

NM and CM callable (difference noted below).

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

### Syntax

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

### Functional Return

*entrynum*      **16-bit signed integer (assigned functional return)**

Returns the command entry number. If the *dictionary* parameter is not specified, 0 is returned.

### Parameters

*cmdimage*      **character array (required)**

A *cmdimage* must begin with an alphabetic character. The first non-alphanumeric character (or space) encountered terminates the command name. Whatever follows is interpreted as an element of the parameter list.

Passes one of two things:

- Command name (if the *dictionary* parameter is specified), then parameters and a carriage-return character (%15). The command name is delimited by the first nonalphanumeric character and cannot be preceded by any leading blanks. One or more blanks is expected after the command name, preceding the parameter list. The parameters are formatted and referenced in the *params* array and *cmdimage* is converted to uppercase, and the character array specified by *dictionary* is searched for a name matching the command.
- Command parameters (if the *dictionary* parameter is not specified), then a carriage-return character (%15). These parameters are formatted. Leading and trailing blanks are ignored. Lowercase is upshifted.

*delimiters*    **character array (optional)**

Passes a string of up to 32 legal delimiters, each of which is an ASCII special character. The last character must be a carriage return. Each delimiter is identified later by its position in this string.

Default: The delimiter array “,=; (carriage return)”.

*maxparms*      **16-bit signed integer by value (required)**

Passes the maximum number of parameters expected in *cmdimage*. Accepts values from 0 to 8192.

*numparms*    **16-bit signed integer by reference (required)**

Returns the number of parameters found in *cmdimage*.

*params*    **32-bit signed integer array (required)**

Returns an array of *maxparms* entries. *Numparms* entries are returned that delineate the parameters. Each entry consists of two 32-bit words. When the intrinsic is executed, the first *numparms* entries are returned to your process in this array, with the first entry corresponding to the first parameter, the second entry corresponding to the second parameter, and so forth. The parameter fields of *cmdimage* are delimited by the delimiters specified in *delimiters*. The pointer in the first word of each entry points to the parameter in *cmdimage*. The string in *cmdimage* is upshifted. The second word of *params* contains the delimiter number and parameter information. Each word in the array named by *params* contains the following information:

- Word 1: The pointer to the first character of the parameter. If the parameter is empty or all blanks, the pointer indicates the location of the delimiter.
- Word 2: Bits that describe the parameter:

Bits	Value/Meaning				
27:5	The zero-relative position of the delimiter in <i>delimiters</i> . For example, if you use the default <i>delimiters</i> array and the current parameter is delimited by a semicolon, this field contains 2.				
26:1	Special characters indicator bit (set by MYCOMMAND): <table style="margin-left: 2em;"> <tr> <td style="vertical-align: top;">0</td> <td>The parameter contains no special characters.</td> </tr> <tr> <td style="vertical-align: top;">1</td> <td>The parameter contains special characters other than those listed in <i>delimiters</i>.</td> </tr> </table>	0	The parameter contains no special characters.	1	The parameter contains special characters other than those listed in <i>delimiters</i> .
0	The parameter contains no special characters.				
1	The parameter contains special characters other than those listed in <i>delimiters</i> .				

## MYCOMMAND

Bits	Value/Meaning
25:1	Numeric character indicator: 0 The parameter does not contain numeric characters. 1 The parameter contains numeric characters.
24:1	Alphabetic characters indicator: 0 The parameter does not contain alphabetic characters. 1 The parameter contains alphabetic characters.
16:8	The length of the parameter in bytes. This value is zero if you omit the parameter.
0:16	Always zero.

### *dictionary* **character array (optional)**

Passes the character array that is to be searched for the command name in *cmdimage*. The format must be identical to that of the *dictionary* parameter in the SEARCH intrinsic. The command, delimited by a blank, is extracted from *cmdimage*, and the SEARCH intrinsic is called with the command name used as the *buffer* parameter of SEARCH. If the command name is found in *dictionary*, the functional return is its entry number. If the command is not found, or if the *dictionary* parameter is not specified, the functional return is zero. If *dictionary* is specified, but the command name is not found in *dictionary*, the parameters specified in *cmdimage* are not formatted.

If this parameter is omitted, the functional return is 0.

### *definition* (NM) **32-bit address (optional)** (CM) **16-bit address (optional)**

Returns a pointer to the byte following the command in *dictionary*. This should be the *definition*, but may be invalid if *definition* is not used for a command. Check the return status before the pointer is used.



## Operation Notes

The MYCOMMAND intrinsic analyzes the command string, identifying the location and characteristics of each parameter. If the dictionary is specified, the first parameter is considered a command name. The array specified in *dictionary* is searched for the command name. This dictionary is specially formatted as a SEARCH intrinsic array. (Refer to the SEARCH intrinsic for the format of the array.) The entry number for the match is returned in *entrynum*. If a definition exists for the matching entry, its address is returned in *definition*.

All remaining entries in the command string are analyzed as parameters. (If *dictionary* is not entered, all input is considered parameter data.) A parameter consists of all characters from the current pointer position to the next delimiter. A pointer to first character is determined. Valid delimiters are defined in *delimiters* or by the default delimiter “,=(carriage return)”. The delimiter type is identified by its position in the defined delimiter list, for example, a comma is identified as 1, and a semicolon as 3. The contents of the parameter are identified as special characters, numeric, or alphanumeric characters.

Results of the analysis are returned in the array *params*. Two words of information are specified for each parameter. The location pointer is entered in word 1, and the delimiter and character information are entered in word 2 for each parameter. A maximum of *maxparms* parameters are analyzed. The actual number of parameters found are returned to *numparms*. The *params* array can then be referred to in subsequent commands to access the parameter data.

If the number of characters in *cmdimage* exceeds 255 characters and no delimiter is present, the calling process is aborted and the following error message is printed:

```
PARSED PARAM OF COMIMAGE > 255 CHARACTERS
```

## Condition Codes

- CCE (2) Request granted. The parameters were formatted, without exception. If *dictionary* was specified, the functional return is the command entry number.
- CCG (0) Request granted. More parameters were found in *cmdimage* than were allowed by *maxparms*. Only the first *maxparms* of these parameters were formatted in *params* and returned.
- CCL (1) Request denied. The *dictionary* parameter was specified, but the command name was not located in the array *dictionary*. The parameters in *cmdimage* were not formatted.

## Related Information

Intrinsics	SEARCH
Commands	None
Manuals	None

---

## 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.

### Syntax

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

### Parameters

*formaldesig* **character array (required)**

Passes and returns a formal file designator. When passed, the file name must end with three blanks. When returned, the language ID has been appended to the name.

*langnum* **16-bit signed integer by value (required)**

Contains the language ID number, specifying the catalog to be opened.

*error* **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid file name
4	File name not terminated by three blanks
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

### Related Information

Intrinsics      CATCLOSE, CATOPEN, CATREAD

Commands        None

Manuals         *Native Language Programmer's Guide (32650-90022)*

## NLCOLLATE

NM and CM callable.

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

### Syntax

```

      CA      CA      I16V      I16
NLCOLLATE(buffer1,buffer2,bufferlength,result,
          I16V      U16A      U16A
          langnum,error,collseq);

```

### Parameters

- buffer1*      **character array (required)**  
 Passes the first character string to be collated.
- buffer2*      **character array (required)**  
 Passes the second character string to be collated.
- bufferlength*    **16-bit signed integer by value (required)**  
 Passes the length (in bytes) of *buffer1* and *buffer2*.
- result*        **16-bit signed integer by reference (required)**  
 Returns the result of the collated character strings (*buffer1* and *buffer2*):

Value	Meaning
0	<i>buffer1</i> collates equal to <i>buffer2</i>
-1	<i>buffer1</i> collates before <i>buffer2</i>
1	<i>buffer1</i> collates after <i>buffer2</i>

The result is 0 if a nonzero error is returned.

- langnum*      **16-bit signed integer by value (required)**  
 Contains the language ID number, specifying the collating sequence to be used.

## NLCOLLATE

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid collating table entry
4	Invalid <i>bufferlength</i> parameter
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

*collseq*      **16-bit unsigned integer array (optional)**

Passes the native language collating sequence table as returned by *itemnum*=11 of the NLINFO intrinsic. If this parameter is present, *langnum* is ignored, and this routine is more efficient. Refer to *itemnums* 11 and 27 of the NLINFO intrinsic.

### Related Information

Intrinsics      NLFINDSTR, NLJUDGE, NLKEYCOMPARE, NLREPCHAR, NLSCANMOVE,  
NLSWITCHBUF, NLTRANSLATE

Commands      None

Manuals      *Native Language Programmer's Guide (32650-90022)*

## NLCOLLATE2

NM callable.

Collates two character strings according to the specified language collating sequence and determines a lexical ordering. Designed to be used with a COBOL program.

### Syntax

```

I32          CA      I32V CA      I32V
result:=NLCOLLATE2(buffer1,len1,buffer2,len2,
                  I16V   U16A  U16A U16A
                  langnum,error,collseq);
```

### Parameters

***result***      **32-bit signed integer by reference (required)**  
Returns the result of the collated character strings (*buffer1* and *buffer2*):

Value	Meaning
0	<i>buffer1</i> collates equal to <i>buffer2</i>
-1	<i>buffer1</i> collates before <i>buffer2</i>
1	<i>buffer1</i> collates after <i>buffer2</i>

The result is 0 if a nonzero error is returned.

***buffer1***      **character array (required)**

Passes the first character string to be collated.

***len1***          **32-bit signed integer by value (required)**

The length, in bytes, of the first character string.

***buffer2***      **character array (required)**

Passes the second character string to be collated.

***len2***          **32-bit signed integer by value (required)**

The length, in bytes, of the second character string.

***langnum***      **16-bit signed integer by value (required)**

Contains the language ID number, specifying the collating sequence to be used.

## NLCOLLATE2

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid collating table entry
4	Invalid <i>bufferlength</i> parameter
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

*collseq*      **16-bit unsigned integer array (optional)**

Passes the native language collating sequence table as returned by *itemnum*=11 of the NLINFO intrinsic. If this parameter is present, *langnum* is ignored, and this routine is more efficient. Refer to *itemnums* 11 and 27 of the NLINFO intrinsic.

### Related Information

Intrinsics      NLFINDSTR, NLJUDGE, NLKEYCOMPARE, NLREPCHAR, NLSCANMOVE,  
                  NLSWITCHBUF, NLTRANSLATE

Commands      None

Manuals      *Native Language Programmer's Guide (32650-90022)*

## 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.

### Syntax

```

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

### Functional Return

*time*            **32-bit signed integer (assigned functional return)**

Returns the time in the following format:

Bits	Value/Meaning
8:8	Minute of hour
0:8	Hour of day

### Parameters

*buffer*            **character array (required)**

Passes the formatted time to be converted.

*bufferlength*    **16-bit signed integer by value (required)**

Passes the length of *buffer* (in bytes).

*langnum*           **16-bit signed integer by value (required)**

Contains the language ID number, specifying the custom time format for *buffer*.

## NLCONVCLOCK

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid time string
4	Invalid length
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

### Related Information

Intrinsics      NLCONVCUSTDATE, NLFMTCALNDAR, NLFMTCLOCK, NLFMTCUSTDATE,  
NLFMTDATE, NLFMTLONGCAL

Commands      None

Manuals      *Native Language Programmer's Guide (32650-90022)*



## 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.

### Syntax

```

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

```

### Functional Return

*date*            **16-bit unsigned integer (assigned functional return)**

Returns the date in the following format:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

### Parameters

*buffer*            **character array(required)**

Passes the formatted date to be converted into custom date format. Leading and trailing blanks are ignored.

*bufferlength*    **16-bit signed integer by value (required)**

Contains the length of *buffer* (in bytes).

*langnum*           **16-bit signed integer by value (required)**

Contains the language ID number, specifying the custom date format for *buffer*.

## NLCONVCUSTDATE

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid date string
4	Invalid string length
5	*NLS internal error
6	*NLS internal error
7	Separator character in <i>buffer</i> does not match custom date template
8	Length of date string > 13 characters (excluding leading and trailing blanks)

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

---

### Note

When a byte array is passed the code receives the starting position and the length. There is no way for the code to determine if the length you specify is correct. Therefore, an error of 4 (invalid string length) is returned only if the specified buffer length is 0 or a negative number. If the passed array length is greater than zero, but is still incorrect, unpredictable results can occur.

---

### Related Information

Intrinsics	NLCONVCLOCK, NLFMTCALENDAR, NLFMTCLOCK, NLFMTCUSTDATE, NLFMTDATE, NLFMTLONGCAL
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

## 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.

### Syntax

```

      I16V      CA      I16V      CA
NLCONVNUM(langnum,instring,inlength,outstring,
      I16V  U16V  U16V
      outlength,error,numspec,
      U16V  U16V      0-V
      fmtmask,decimals);

```

### Parameters

- langnum*     **16-bit signed integer by value (required)**  
 Contains the language ID number, specifying which numeric formatting rules are to be used in the conversion.
- instring*    **character array (required)**  
 Contains the native language formatted number to be converted. Leading and trailing spaces are ignored.
- inlength*    **16-bit signed integer by value (required)**  
 Passes the length, in bytes, of *instring*.
- outstring*    **character array (required)**  
 Contains the converted output, left justified in the buffer. *Outstring* can reference the same address as *instring*.
- outlength*   **16-bit signed integer by value (required)**  
 Passes the length, in bytes, of *outstring*. If the call is successful, *outlength* contains the actual length of the converted number.

## NLCONVNUM

*error*      **16-bit unsigned integer by value (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid length specified ( <i>inlength</i> or <i>outlength</i> )
4	Invalid number specified ( <i>instring</i> )
5	*NLS internal error
6	*NLS internal error
7	Truncation occurred ( <i>outstring</i> partially formatted)
8	Invalid <i>numspec</i> parameter
9	Invalid <i>fmtmask</i> parameter

\* Do not apply to calls with *langnum* equal to 0 (NATIVE3000/XL).

*numspec*      **16-bit unsigned integer by value (optional)**

Returns a byte array from NLNUMSPEC, containing format information. If this parameter is present, *langnum* is ignored and performance is improved (refer to NLNUMSPEC).

*fmtmask* **16-bit unsigned integer by value (optional)**

Specifies how to format the number. The default value is 0 and indicates substitution only.

Bits	Value/Meaning
15:1	0 Convert thousands separators
	1 Strip thousands separators
14:1	0 Convert decimal separators
	1 Strip decimal separators
13:1	0 Any character can be in <i>instring</i> (no validation as a number is performed)
	1 Number contained in <i>instring</i> (no validation as a number is performed)
0:13	Reserved for the operating system

*decimals* **16-bit unsigned integer by value (optional)**

If *fmtmask* bit 13 is set to 1, *decimals* returns the number of decimal places in the input number.

**Operation Notes**

Split-stack calls are not permitted.

This intrinsic either:

- Converts a native language formatted number to an ASCII number with the NATIVE-3000 decimal separator (.) and thousands separator (,) for use in further conversion to INTEGER, REAL, and so on.
- Converts the decimal and thousands separators, or strips them (see *fmtmask*), to the NATIVE-3000 equivalent.

For languages using an alternate set of digits (ARABIC and HINDI digits only), this intrinsic converts the digits to ASCII for recognition and use as numeric characters.

**Related Information**

Intrinsics      NLFMTNUM, NLNUMSPEC, NLSUBSTR

Commands        None

Manuals         *Native Language Programmer's Guide* (32650-90022)

---

## NLFINDSTR

NM and CM callable.

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

### Syntax

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

### Functional Return

*offset*            **16-bit signed integer (assigned functional return)**  
Returns a -1 if *string2* is not found in *string1*.

### Parameters

*langnum*            **16-bit signed integer by value (required)**  
Contains the language ID number.

*string1*            **character array (required)**  
Contains the string of characters to be searched. It can contain 1 byte and 2 byte Asian characters.

*length1*            **16-bit signed integer by value (required)**  
Passes the length, in bytes, of *string1*.

*string2*            **character array (required)**  
Contains the character string to be searched for.

*length2*            **16-bit signed integer by value (required)**  
Passes the length, in bytes, of *string2*.

***error*** 16-bit unsigned integer array (required)

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid <i>length1</i> parameter
4	Invalid <i>length2</i> parameter
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with *langnum* equal to 0 (NATIVE3000/XL).

***charset*** 16-bit unsigned integer array (optional)

Contains the character set definition for the language to be used, as returned by *itemnum*= 12 of the NLINFO intrinsic. When specifying a character set, be sure that the language ID used matches the character set.

**Operation Notes**

Split-stack calls are not permitted.

**Related Information**

Intrinsics	NLCOLLATE, NLJUDGE, NLKEYCOMPARE, NLREPCHAR, NLSCANMOVE, NLSWITCHBUF, NLTRANSLATE
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## 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.

### Syntax

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

### Parameters

*date*           **16-bit unsigned integer by value (required)**

Contains the date format returned by the CALENDAR intrinsic:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*buffer*       **character array (required)**

Returns the formatted date. This string is 18 characters long and is padded with blanks (if necessary). Using a buffer of a different size may produce unexpected results.

*langnum*      **16-bit signed integer by value (required)**

Contains the language ID number, specifying the calendar template to be used. A *langnum* of 0 returns the date formatted as if FMTCALENDAR were used (for example, FRI, OCT 1, 1982).



**error**      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid date value
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

**Related Information**

Intrinsics	NLCONVCLOCK, NLCONVCUSTDATE, NLFMTCLOCK, NLFMTCUSTDATE, NLFMTDATE, NLFMTLONGCAL
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## NLFMTCLOCK

NM and CM callable.

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

### Syntax

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

### Parameters

*time*            **32-bit signed integer by value (required)**

Contains the time format returned by the `CLOCK` intrinsic:

Bits	Value/Meaning
24:8	Tenths of seconds
16:8	Seconds
8:8	Minute of hour
0:8	Hour of day

*buffer*        **character array (required)**

Returns the formatted time of day in an 8-character array. Using a buffer of a different size may produce unexpected results.

*langnum*      **16-bit signed integer by value (required)**

Contains the language ID number, specifying which format to use. A *langnum* of 0 returns the time formatted as if you used `FMTCLOCK`.

**error**      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid time format
4	*NLS internal error
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

**Related Information**

Intrinsics	NLCONVCLOCK, NLCONVCUSTDATE, NLFMTCALENDAR, NLFMTCUSTDATE, NLFMTDATE, NLFMTLONGCAL
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## 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.

### Syntax

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

### Parameters

*date*            **16-bit unsigned integer by value (required)**

Contains the date format returned by the `CALENDAR` intrinsic:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*buffer*            **character array (required)**

Returns the formatted date in a 13-character array. Using a buffer of a different size may produce unexpected results.

*langnum*           **16-bit signed integer by value (required)**

Contains the language ID number, specifying the custom date template to be used for formatting. A *langnum* of 0 returns the time formatted as if `FMTCLOCK` were used.

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid date value
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

### Related Information

Intrinsics	NLCONVCLOCK, NLCONVCUSTDATE, NLFMTCALENDAR, NLFMTCLOCK, NLFMTDATE, NLFMTLONGCAL
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## NLFMTDATE

NM and CM callable.

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

### Syntax

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

### Parameters

*date*      **16-bit unsigned integer by value (required)**

Contains the date format returned by the CALENDAR intrinsic:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*time*      **32-bit signed integer by value (required)**

Contains the time to be formatted, returned by the CLOCK intrinsic:

Bits	Value/Meaning
8:8	Minutes of hour
0:8	Hour of day

*buffer*    **character array (required)**

Returns the formatted date and time in a 28 byte array. Using a buffer of a different size may produce unexpected results.

*langnum*   **16-bit signed integer by value (required)**

Contains the language ID number, specifying the formatting templates to use. A *langnum* of 0 returns the date/time string as if FMTDATE were used (for example, MON, FEB 7, 1983 9:00 AM).

**error**      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid date value
4	Invalid time value
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

**Related Information**

Intrinsics	NLCONVCLOCK, NLCONVCUSTDATE, NLFMTCALENDAR, NLFMTCLOCK, NLFMTCUSTDATE, NLFMTLONGCAL
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## 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`.

### Syntax

LV    BA    IV    LA
NLFMTLONGCAL( <i>date,string,langnum,error</i> )

### Parameters

*date*            **logical by value (required)**

Contains the date format returned by the `CALENDAR` intrinsic:

Bits	Value/Meaning
7:9	Day of year
0:7	Year of century

*string*            **byte array (required)**

Contains the formatted long calendar date, padded with blanks if necessary.

*langnum*            **integer by value (required)**

Contains the language ID number, specifying which format to use.



**error**      **logical array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid date format
4	*NLS internal error
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

**Operation Notes**

Split-stack calls are not permitted.

**Related Information**

Intrinsics	NLCONVCLOCK, NLCONVCUSTDATE, NLFMTCALENDAR, NLFMTCLOCK, NLFMTCUSTDATE, NLFMTDATE
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## 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.

### Syntax

```
          I16V      CA      I16V      CA
NLFMTNUM(langnum,instring,inlength,outstring,
          I16V      U16A  U16A      U16V  I16V  0-V
          outlength,error,numspec,fmtmask,decimals)
```

### Parameters

- langnum***     **16-bit signed integer by value (required)**  
Contains the language ID number, specifying which format to use.
- instring***     **character array (required)**  
Contains the NATIVE-3000 formatted ASCII number to be converted (for example, \$123,456.78). Leading and trailing blanks are allowed.
- inlength***     **16-bit signed integer by value (required)**  
Contains the length, in bytes, of *instring*.
- outstring***    **character array (required)**  
Contains the returned language-specific formatted number. The decimal separator, thousands separator, and currency symbol/name are replaced (if present), or are inserted (if specified by *fmtmask*), according to the language definition. The *outstring* can reference the same address as *instring*.
- outlength***    **16-bit signed integer by value (required)**  
Contains the length, in bytes, of *outstring*. After a successful call, if *outstring* is returned left-justified (specified by *fmtmask*), *outlength* returns the actual length, in characters, of the formatted number.

*error***16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid length specified ( <i>inlength</i> or <i>outlength</i> )
4	Invalid number specified ( <i>instring</i> )
5	*NLS internal error
6	*NLS internal error
7	Truncation occurred ( <i>outstring</i> partially formatted)
8	Invalid <i>numspec</i> parameter
9	Invalid <i>fmtmask</i> parameter
10	Invalid <i>decimals</i> parameter

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

*numspec***16-bit unsigned integer array (optional)**

Contains format specifications, returned from NLNUMSPEC, for the specified language (currency/name, decimal separator, and so on). If this parameter is present, *langnum* is ignored, and performance is improved.

## NLFMTNUM

*fmtmask*

### 16-bit unsigned integer by value (optional)

Contains formatting specifications for the input. The default value is 0, a simple substitution.

Bits	Value/Meaning
15:1	Thousands separators: 0 Do not insert thousands separators 1 Insert thousands separators
14:1	Decimal separators: 0 Do not insert decimal separators 1 Insert decimal separators
13:1	Currency symbol/name: 0 Do not insert currency symbol/name 1 Insert currency symbol/name
11:2	Output justification: 0 Do not justify output 1 Left-justify output 2 Right-justify output 3 Left-justify output and return the formatted number length in <i>outlength</i>
0:11	Reserved for the operating system

*decimals*

### 16-bit signed integer by value (optional)

Specifies where to insert the decimal separator. The value is ignored if bit 14 of *fmtmask* is zero or if a decimal separator is present in the number.

## Operation Notes

Split-stack calls are not permitted.

This intrinsic operates in substitution mode and formatting mode:

Substitution mode	If <i>fmtmask</i> is omitted or has all bits set to zero, the substitution mode substitutes the native equivalent for "." and ","; for ARABIC, it substitutes the alternative set of digits for ASCII digits. The input is not validated as a number and can contain several numbers. No justification takes place, and the output is left-truncated if <i>outstring</i> is shorter than <i>instring</i> (for example, 1,234.56 becomes ,234.56).
Formatting mode	If any bit 10-15 in <i>fmtmask</i> is set to one, the formatting performs the substitution and formats the input according to <i>fmtmask</i> . In this mode, input is validated as a number, and only ASCII digits and ".", ",", "-", "+", and "\$" are allowed. Only one sign and one "\$" are allowed. They must be the character(s) in <i>instring</i> . Even if insertion (of the separators) is specified in <i>fmtmask</i> , the thousands and decimal separators are still valid characters in the input. In this case, they are substituted. If no justification is specified, the output is right-justified with the same number of trailing spaces as the input. If the output is truncated, it is left-truncated.

---

**Note** For languages written right-to-left, trailing spaces in the input are preserved as leading spaces in the output.

---

## Related Information

Intrinsics	NLCONVNUM, NLNUMSPEC
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## 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.

### Syntax

I16	I16V	U16A
<i>langnum</i> := NLGETLANG( <i>langtype</i> , <i>error</i> );		

### Functional Return

*langnum*      **16-bit signed integer (assigned functional return)**

Returns the language ID number of the current user, data, or system. If an error occurs, *langnum* returns a value of 0 (NATIVE3000/XL).

### Parameters

*langtype*      **16-bit signed integer by value (required)**

Contains the function number indicating which type of language ID number is returned. The possible values are:

Value	Meaning
1	The user-interface language; specifies the language used for communication between the program and user
2	The data language; determines how the subsystem performs various language-dependent data manipulation functions (for example, sorting, upshifting)
3	The system default language

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	NLS not installed
2	Found language requested, but not configured on system
3	Invalid <i>langtype</i> value
4	No language specified

### Operation Notes

The NLGETLANG intrinsic locates the language ID numbers requested by *langtype* 1 and 2 by referring to the Hewlett-Packard defined job control words (JCWs), NLUSERLANG and NLDATA LANG respectively. If the required JCW does not exist or has a value greater than or equal to FATAL (32768), The error value of 4 is returned.

### Related Information

Intrinsics	ALMANAC, NLINFO
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## 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

### Syntax

I16V   *   I16   U16A NLINFO( <i>itemnum</i> , <i>item</i> , <i>langnum</i> , <i>error</i> );
--

### Parameters

*itemnum*    **16-bit signed integer by value (required)**

Contains the *item* to be returned.

*item*        **type of variable depends on itemnum (required)**

Returns information requested or the language name or number requested. Table 4-24 lists the defined *itemnums*, data types (mnemonics), and a description of the information returned to *item*.

*langnum*    **16-bit signed integer by reference (required)**

Contains the language ID number, for the information requested, except for NLINFO *itemnums* 22 and 24. For NLINFO *itemnum*= 22, the language ID number is returned in *langnum*; for NLINFO *itemnum*= 24, the character set ID number is returned in *langnum*.



*error***16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

<b>Value</b>	<b>Meaning</b>
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	*Specified character set not configured
4	No national table present
5	*NLS internal error
6	*NLS internal error
7-9	Reserved for the operating system
10	<i>Itemnum</i> out of range

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

**Table 4-24. NLINFO Itemnum/Item Values**

Itemnum	Mnemonic	Item Description
1	CA	<p>Returns the calendar format in an 18 byte array. The 18 bytes of the string for this definition are interpreted as the format description for that language. The following descriptors are valid:</p> <p>D        1 byte day abbreviation            DD      2 byte day abbreviation            DDD    3 byte day abbreviation            M       1 byte month abbreviation            MM     2 byte month abbreviation            MMM    3 byte month abbreviation            MMMM 4 byte month abbreviation            mm     Numeric month of the year            dd     Numeric day of the month            yy     Numeric year of the century            yyyy   Numeric year            Nyy    National year            NPy    National year that can include a before-period symbol</p> <p>E        1-8 of these are replaced by an equal amount of bytes from the emperor/country name</p> <p>Valid separators are any special character. For example, a format can be <b>DDD, MMM DD, yyyy</b>. Using this format in <b>NATIVE3000/XL</b> results in <b>FRI, MAY 25, 1984</b>.</p>

Table 4-24. NLINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
2	CA	<p>Returns the custom date format in a 13 byte array. The 13 bytes of the string for this definition are interpreted as the custom date format description. The following descriptors are valid:</p> <p>mm      Numeric month of the year  dd      Numeric day of the month  yy      Numeric year of the century  yyyy    Numeric year  Nyy     National year  NPyy    National year that can include a before-period symbol</p> <p>Valid separators are any special character. For example, a date format can be <code>yy/mm/dd</code>. This format in <code>NATIVE3000/XL</code> results in <code>81/03/25</code>.</p>
3	CA	<p>Returns the clock specification in an 8 byte array. This 8 byte string provides the clock format description (template) <code>HHSXXYYZ</code>, where the elements specify the following:</p> <p>HH      Clock hour specification, either ' ' or '\$'  S        Separator; valid separators can be any special or alphabetic character, or 0 for no separator between hours and minutes  XX      Symbol for AM  YY      Symbol for PM  Z        Suppresses leading zero (of hours) if blank; prints leading zero if 0</p> <p>In suppression of leading zero, " " (leading zero suppressed) or 0 (leading zero is printed) are valid. For example, the format <code>"12:AMPM "</code> results in the formatted clock information <code>9:06 AM</code>; the leading zero is suppressed. If the clock specification were changed to <code>"240 0"</code>, the formatted clock information for the same time is: <code>0906</code>. Note the four blanks used as place holders to ensure the correct placement of the leading-zero suppression character.</p>
4	CA	<p>Returns the month abbreviation table in a 48 byte array. Each abbreviation is 4 bytes long, using blank padding where necessary to maintain uniform length in all native language abbreviations. For example, the <code>NATIVE3000/XL</code> abbreviations contain 3 bytes plus a blank. The first 4 bytes of the array contain the abbreviation of January. For example, the month abbreviation table is:  <code>"JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC "</code></p>
5	CA	<p>Returns the month table in a 144 byte array. Each month's name can be up to 12 bytes long. Unused space in the month name is padded with blanks where necessary to equal 12 bytes. The table begins with the language-dependent equivalent in the native language specified for January. For example, the month name table is:  <code>"JANUARY FEBRUARY MARCH ... DECEMBER"</code></p>
6	CA	<p>Returns the day abbreviation table in a 21 byte array. Each abbreviation is 3 bytes long. The table begins with Sunday. For example, the day abbreviation table is:  <code>"SUNMONTUEWEDTHUFRISAT"</code></p>

Table 4-24. NLINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
7	CA	Returns the table containing the day of the week in an 84 byte array. Each day is 12 bytes long (with blank padding as needed). The table starts with Sunday. For example, the day name table is: "SUNDAY MONDAY TUESDAY ... SATURDAY"
8	CA	Returns the YES/NO responses in a 12 byte array; the first 6 bytes contain the (upshifted) YES response; the second 6 bytes contain the (upshifted) NO response.
9	CA	Returns the symbols for decimal separator and thousands indicator in a 2 byte array. The first byte contains the decimal separator, the second byte contains the thousands indicator. The second byte can take a special value of 0. Do not take this value literally as a thousands separator; it signifies the absence of a thousands separator for the language chosen.
10	CA	Returns the currency signs in a 6 byte array. The first byte represents the short currency symbol (if any) used for business formats; the second byte is a flag indicating whether the currency symbol precedes or succeeds the number and whether the currency symbol is preceded or succeeded by blanks. The last four bytes contain the full currency symbol. The layout of the second byte is:  Bits 0:4 0 - Currency symbol has no blanks preceding or succeeding it. 1 - Currency symbol has a blank preceding it. 2 - Currency symbol has a blank succeeding it. 3 - Currency symbol has blanks preceding and succeeding it.  Bits 4:4 0 - Currency symbol precedes the number. 1 - Currency symbol succeeds the number. 2 - Currency symbol replaces the decimal separator.
11	U16A	Returns the collating sequence table. A call to <b>NLINFO</b> <i>itemnum</i> =27 determines the length of this array based on the length of the table of the native language specified.
12	CA	Returns the character set attribute table in a 256-element (256-half word) array. Each character contains the numeric identification of the character type:  0        Numeric character 1        Alphabetic lowercase character 2        Alphabetic uppercase character 3        Undefined graphic character 4        Special character 5        Control code
13	CA	Returns the ASCII-to-EBCDIC translation table in a 256 byte array.
14	CA	Returns the EBCDIC-to-ASCII translation table in a 256 byte array.
15	CA	Returns the upshift table in a 256 byte array.
16	CA	Returns the downshift table in a 256 byte array.

Table 4-24. NLINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
17	U16A	Returns the language numbers of all configured languages. The first element of this array contains the number of configured languages. The second element contains the language number of the first configured language. The third element contains the language number of the second configured language, and so forth. (The <i>langnum</i> parameter is disregarded.)
18	I16	Returns -1 if the specified language is supported (configured) on the system. Otherwise, 0 is returned.
19	I16	Returns the character set ID number supporting the specified language.
20	CA	Returns the uppercase name of the character set supporting the specified language in a 16 byte array. If the name contains fewer than 16 bytes, it is padded with blanks.
21	CA	Returns the uppercase name of the specified language in a 16 byte array. If the name contains fewer than 16 bytes, it is padded with blanks.
22	CA	Passes a language name or number (in ASCII digits) terminated by a blank. The array can be <= 16 bytes. The associated language ID number is returned to <i>langnum</i> .
23	I16	Returns -1 if the character set specified is supported (configured) on the system. Otherwise, 0 is returned.
24	CA	Passes a character set name or number (in ASCII digits) terminated by a blank. The array can be <= 16 bytes. The associated character set ID number is returned to <i>langnum</i> .
25	CA	Returns the uppercase name of the specified character set in a 16 byte array. The <i>langnum</i> parameter must contain the ID number of the character set. If the name contains fewer than 16 bytes, it is padded with blanks.
26	I16	Returns the class number of the specified language.
27	I16	Returns the length (in half words) of the collating sequence table of the specified language (see <i>itemnum</i> = 11).
28	I16	Returns the length (in half words) of the national-dependent information table. If no national table exists for the specified language, <i>error</i> =4 is returned.
29	U16A	Returns the national-dependent information table. To determine the size of this array, you must first obtain the length with a call to <b>NLINFO</b> <i>itemnum</i> =28.

**Table 4-24. NLINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
30	CA	<p>A 36 byte array where the long calendar format is returned. It can contain arbitrary text and the following descriptors:</p> <p>D      1-3 of these are replaced by an equal number of bytes from the day abbreviation.</p> <p>W      1-12 of these are replaced by an equal number of bytes from the day of week.</p> <p>dd     Numeric day of month.</p> <p>M      1-4 of these are replaced by an equal number of bytes from the month abbreviation.</p> <p>O      1-12 of these are replaced by an equal number of bytes from the month of year.</p> <p>mm     Numeric month of year.</p> <p>yy     Numeric year of century.</p> <p>Nyy    National year.</p> <p>NPy    National year that can include a before-period symbol.</p> <p>E      1-8 of these are replaced by an equal number of bytes from the emperor/country name.</p> <p>A literal character (~) can be used to indicate that one of the above special characters are taken literally within a format. For example, a format of <code>WWWWWWW, 00000000 dd, A.~D.yyyy</code> results in <code>WEDNESDAY, NOVEMBER 21, A.D. 1984</code>.</p>
31	CA	<p>A 16 byte array where the currency name is returned.</p>
32	CA	<p>An 8 byte array containing information about an alternate set of digits (Arabic only).</p> <p>0-1    Alternative digit separator (integer)                0 - No alternative digits defined                1 - Alternative digits defined</p> <p>2      Alternative digit 0</p> <p>3      Alternative digit 9</p> <p>4      "+" used with alternative digits</p> <p>5      "." used with alternative digits</p> <p>6      Decimal separator used with alternative digits</p> <p>7      Thousands separator used with alternative digits</p>

Table 4-24. NLINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
33	CA	A 4 byte array containing information about the direction of the language. 0-1      Language direction (integer) 0 - Direction is left-to-right 1 - Direction is right-to-left 2         The right-to-left space 3         Undefined
34	U16	A logical value that returns the data ordering of the language. 0         Keyboard order 1         Left-to-right screen order 2         Right-to-left screen order
35	U16	A logical value that returns the language character size. 0         1 byte characters (8-bits) 1         2 byte characters (16-bits)
36	U16	A logical value that returns a true (1), if the language requires suppressing the leading zero or a blank in the date format.

**Related Information**Intrinsics      **ALMANAC**, **NLGETLANG**

Commands      None

Manuals        *Native Language Programmer's Guide (32650-90022)*

---

## NLJUDGE

NM and CM callable.

Judges whether a character is a 1 byte or 2 byte Asian character.

### Syntax

```
I16V          I16V   CA      I16V
n2bytes:=NLJUDGE(langnum,instring,stringlength,
                 CA U16A U16A
                 flags,error,charset);
```

### Functional Return

*n2bytes*      **16-bit signed integer (assigned functional return)**  
Contains the number of 2 byte Asian characters in a 16-bit signed integer value that can be used to check a string of characters for Asian characters.

### Parameters

*langnum*      **16-bit signed integer by value (required)**  
Contains the language ID number.

*instring*      **character array (required)**  
Contains the string of characters to be judged.

*stringlength*      **16-bit signed integer by value (required)**  
Specifies the number of bytes in *instring*.

*flags*          **character array (required)**  
Contains one of the following values (ASCII characters):

Value	Meaning
0	1 byte character
1	First byte of a 2 byte character
2	Second byte of a 2 byte character
3	Invalid Asian character



*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language is not configured
3	Invalid string length
4	Not returned
5	*NLS internal error
6	*NLS internal error
7	*Invalid characters found in <i>instring</i>

\* Do not apply to calls with *langnum* equal to 0 (NATIVE-3000).

*charset*      **16-bit unsigned integer array (optional)**

Contains the character set definition for the language to be used, as returned by *itemnum*=12 of NLINFO.

### Related Information

Intrinsics	NLCOLLATE, NLKEYCOMPARE, NLREPCHAR, NLSCANMOVE, NLSWITCHBUF, NLSTRANSLATE
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## NLKEYCOMPARE

NM and CM callable.

Compares two strings of different length (for use with KSAM generic key searching).

### Syntax

```
CA      I16V  CA  I16V
NLKEYCOMPARE(generickey,length1,key,length2,
             I16  I16V  U16A U16A
             result,langnum,error,collseq);
```

### Parameters

- generickey***    **character array (required)**  
Passes the generic key to be compared to the keys contained in the record read by FREAD.
- length1***      **16-bit signed integer by value (required)**  
Contains the length, in bytes, of *generickey*, which must be less than *length2*.
- key***            **character array (required)**  
Passes an entire key to which you want to compare *generickey*.
- length2***      **16-bit signed integer by value (required)**  
Contains the length in bytes of *key*, which must be greater than *length1*.
- result***        **16-bit signed integer by reference (required)**  
Returns the result of the compare:

Value	Meaning
0	The retrieved key matches the generic key exactly for a length of <i>length</i> .
1	The retrieved key does not match the generic key; it is different only because of priority (for example, uppercase versus lowercase characters or accent). The <b>FREAD</b> key is still in range; records can follow whichever key matches the generic key exactly.
2	The retrieved key is less than the generic key (its collating order precedes the key specified). It does not match <i>generickey</i> . This means the <b>FREAD</b> call found a record that precedes the range requested. Records that match <i>generickey</i> can follow.
3	The retrieved key is greater than the generic key (it collates after the specified key). This means that the <b>FREAD</b> call found a record whose key follows the specified range. No records matching <i>generickey</i> follow.

**langnum**      **16-bit signed integer by value (required)**

Contains the language ID number, specifying the collating sequence to be used for the compare.

**error**            **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid collating table entry
4	Invalid length parameter
5	*NLS internal error
6	*NLS internal error
7	Value of <i>length1</i> not less than <i>length2</i>

\* Do not apply to calls with a **langnum** equal to 0 (NATIVE3000/XL).

**collseq**            **16-bit unsigned integer array (optional)**

Passes the collating sequence table as returned by **NLINFO** item 11. If this parameter is present, **langnum** is ignored, and this routine is much more efficient.

## NLKEYCOMPARE

### Operation Notes

Use `NLKEYCOMPARE` when reading a KSAM file in key sequential order in combination with `FREAD`, and after a `FFINDBYKEY` call.

The `NLKEYCOMPARE` intrinsic lets a program determine whether a generic key search found an exact match (that is, the generic key is exactly equal to the beginning of the key), or whether the keys are different only because of priority (for example, uppercase versus lowercase or accent). It also lets the program determine whether an exactly matching key could be farther along the key sequence.

### Related Information

Intrinsics      `NLCOLLATE`, `NLREPCHAR`, `NLSCANMOVE`, `NLSWITCHBUF`, `NLSTRANSLATE`

Commands      None

Manuals      *Native Language Programmer's Guide* (32650-90022)

## NLMATCH

Compares a string against a pattern which has previously been compiled with the NLMATCHINIT intrinsic.

### Syntax

```

I           BA           BA           IV
result := NLMATCH(comp_pattern,string,length);
```

### Functional Return

Zero is returned in *result* if the *string* matches the compiled *comp\_pattern*. If there is no match, the mismatch location is returned.

<i>result</i>	Integer Error	Meaning
	1	NLS is not installed
	2	The specified language is not configured
	100+n	n is the mismatch location

### Parameters

*comp\_pattern* **Byte array (required)**

A byte array containing the compiled form of a *comp\_pattern* as returned by NLMATCHINIT.

A pattern consists of a series of special characters that indicate the type of data to be selected. These characters are:

*string* **Byte array (required)**

The string to be compared against the *comp\_pattern*.

*length* **Integer by value (required)**

Length, in bytes, of the string to be compared.

### Related Information

Intrinsics     NLMATCHINIT

Commands

Manuals     *Native Language Programmer's Guide* (32650-90022)

---

## NLMATCHINIT

Compiles a pattern into a form for use with the NLMATCH intrinsic.

### Syntax

```
I          BA          I
result := NLMATCHINIT(pattern, pattern_len,
IV         BA          I
langid, comp_pattern, comp_buf_size);
```

### Functional Return

Zero is returned if the compilation of *pattern* is successful. Otherwise the *result* contains the error code.

result	Integer Error	Meaning
	001	NLS is not installed
	002	The specified language is not configured
	101	Missing “]” or “}” operators
	102	Expecting alpha symbols
	104	Incomplete <i>pattern</i> specified
	105	Bad range operation specified
	203	Pattern too big
	900	Internal state machine error

### Parameters

*pattern*      **Byte array (required)**

A byte array containing the pattern to be compiled. Leading and trailing spaces are ignored.

A pattern consists of a series of special characters that indicate the type of data to be selected. These characters are:

Char	Selected
a	Uppercase, lowercase, or 16-bit Asian character
u	Uppercase alphabetic character
l	Lowercase alphabetic character
b	Blank (space)
d	Digits, as defined in the NLS character attribute table
?	Any character

Other metacharacter indicate choice, grouping, and ranges:

Char	Operation
!	Transparency
,	Choice
:	Range
{ }	Grouping
[ ]	Optional
+	Repetition (one or more)
*	Repetition (zero or more)

Pattern operators are evaluated in the following order, where x and y are any character for the language:

Char	Operation	Example
!	Transparency	!x
:	Range	x:y
+/*	Repetition (1/0 or more)	x+ or x*
	Concatenation	xy
,	Choice	x,y

The repetition, concatenation, and choice operators can be used with patterns. For instance: x,y are any pattern in this case.

***pattern\_len*** Integer (required)

Length, in bytes, of the *pattern* to be compiled.

***langid*** Integer by value (required)

The language ID number of the language to be used in compiling the *pattern*.

***comp\_pattern*** Byte array (required)

A byte array specifying where the compiled *pattern* is to be constructed.

***comp\_buf\_size*** Integer (required)

The length of the *comp\_pattern*, in bytes, available for constructing the compiled *pattern*.

## NLMATCHINIT

### Operation Notes

NLMATCHINIT converts the regular expression in *pattern* to a reduced, finite state machine and places it in *comp\_pattern*, where *pattern\_len* is the expression's length in bytes, and *comp\_buf\_size* is the maximum size of the *comp\_pattern* in bytes.

If *comp\_buf\_size* is 0 (zero), only a syntax check of *pattern* is performed and no *comp\_pattern* will be generated.

If NLMATCHINIT returns 0 (zero), the compilation of *pattern* succeeded.

If *comp\_buf\_size* > 0 (zero), the *comp\_pattern* contains the compiled *pattern* and *comp\_buf\_size* contains its actual size in bytes.

If NLMATCHINIT is not zero, an error occurred and NLMATCHINIT returns the error code in return. In such cases, *pattern\_len* contains the zero-origin index into *pattern* and tells you how far the scan has progressed (the scan may have completed). The *pattern* description and grammar are the ones used in VPLUS.

### Related Information

Intrinsics      NLMATCH

Commands

Manuals      *Data Entry and Forms Management System VPLUS Reference Manual* (32209-90001) and *Native Language Programmer's Guide* (32650-90022),



## 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.

### Syntax

```

      I16V   U16A  U16A
NLNUMSPEC(langnum,string,error);
```

### Parameters

*langnum*      **16-bit signed integer by value (required)**

Contains the language ID number.

*string*        **16-bit unsigned integer array (required)**

Contains  $\geq 60$  bytes, where the following information is returned:

Byte(s)	Meaning
00-01	Language identification number (integer)
02-03	Alternate digit indicator (integer)
	0    No alternate digits exist
	1    Alternate digits exist
04-05	Language direction indicator (integer)
	0    Language is left-to-right
	1    Language is right-to-left
06-07	Alternate digit range (0,9)
08	Decimal separator (ASCII-digits)
09	Decimal separator (alternate-digits)
10	Thousands separator (ASCII-digits)
11	Thousands separator (alternate-digits)
12	'+' (alternate-digits)
13	'-' (alternate-digits)
14	Right-to-left space
15	Reserved for the operating system

## NLNUMSPEC

Byte(s)	Meaning
16-17	Currency place (integer)
0	Currency symbol
1	Currency symbol succeeds the number
2	Currency symbol replaces the decimal separator
3	Currency symbol precedes the sign
18-19	Length of currency symbol (integer), includes spaces
20-37	Currency symbol, includes spaces
38-59	Reserved for the operating system

### *error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid string
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with *langnum* equal to 0 (NATIVE3000/XL).

## Operation Notes

Split-stack calls are not permitted.

The intrinsic combines **NLINFO** calls with *itemnums* 9, 10, 31, 32, and 33. The information is formatted where needed (for example, any spaces in the currency symbol/name are included). The currency symbol/name is the shortest nonblank descriptor, as returned from **NLINFO** *itemnums* 10 and 31.

This intrinsic does not provide any information not obtainable with **NLINFO**. It is included for the convenience of the NLS user. For efficiency, the user of this intrinsic calls it only once, saves the result, and calls **NLFMTNUM** and/or **NLCONVNUM** multiple times.

## Related Information

Intrinsics      **NLCONVNUM, NLFMTNUM**

Commands      None

Manuals      *Native Language Programmer's Guide* (32650-90022)

## 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.

### Syntax

```

          CA      CA      I16V
NLREPCHAR(inbuffer,outbuffer,bufferlength,
          CV      I16V  U16A  U16A
          replacechar,langnum,error,charset);
```

### Parameters

*inbuffer*      **character array (required)**  
 Passes the nondisplayable characters to be replaced.

*outbuffer*     **character array (required)**  
 Returns the replaced character string.

*bufferlength* **16-bit signed integer by value (required)**  
 Passes the length, in bytes, of *inbuffer*.

*replacechar* **character by value (required)**  
 Contains the replacement character to be used.

*langnum*       **16-bit signed integer by value (required)**  
 Contains the language ID number.

## NLREPCCHAR

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid replacement character
4	Invalid length parameter
5	*NLS internal error
6	*NLS internal error
7	Invalid <i>charset</i> table entry
8	Overlapping strings, <i>outbuffer</i> overwrites <i>inbuffer</i>

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

*charset*      **16-bit unsigned integer array (optional)**

Passes the character set definition for the language used, as returned in NLINFO *itemnum*= 12. If this parameter is present, *langnum* is ignored, and this intrinsic is more efficient.

### Related Information

Intrinsics      NLCOLLATE, NLKEYCOMPARE, NLSCANMOVE, NLSWITCHBUF, NLSTRANSLATE

Commands      None

Manuals      *Native Language Programmer's Guide* (32650-90022)

---

## NLSCANMOVE

NM and CM callable.

Scans and moves character strings according to character attributes. This function is handled in a language-dependent manner.

### Syntax

I16		CA	CA	U16V
	<i>numchar</i> :=NLSCANMOVE( <i>inbuffer</i> ,	<i>outbuffer</i> ,	<i>flags</i> ,	
		I16V	I16V	U16A
		<i>bufferlength</i> ,		<i>langnum</i> ,
		U16A	CA	
		<i>charset</i> , <i>shiftinfo</i> );		

### Functional Return

*numchar*      **16-bit signed integer (assigned functional return)**  
 Contains the number of characters acted upon in the scan or move function.

### Parameters

*inbuffer*      **character array (required)**  
 Passes the source string.

*outbuffer*      **character array (required)**  
 Returns the target string.

---

**Note**      If *outbuffer* and *inbuffer* are the same string, this intrinsic performs a scan function. Otherwise, a move function is performed.

---

## NLSCANMOVE

**flags** 16-bit unsigned integer by value (required)

Contains the flag defining the options for calling the intrinsic and defines the condition for terminating the scan or move function.

Bits	Value/Meaning
14:2	Alphabetic ( <b>NLINFO</b> item 12, types 1 and 2): 1 Lowercase 2 Uppercase 3 Uppercase/lowercase
13:1	Numeric ( <b>NLINFO</b> <i>itemnum</i> =12, type 0)
12:1	Special; <b>NLINFO</b> <i>itemnum</i> =12, types: 3 Undefined graphic character 4 Special character 5 Control code
11:1	<b>WHILE/UNTIL</b> option, if: 0 String scanned while the condition (specified by <i>flags</i> (12:4)) is true 1 String scanned until the condition (specified by <i>flags</i> (12:4)) is true
9:2	Shift: 1 Upshift 2 Downshift
0:9	Reserved for the operating system

**bufferlength** 16-bit signed integer by value (required)

Passes the maximum number of bytes to be acted upon during the indicated function.

**langnum** 16-bit signed integer by value (required)

Contains the language ID number, specifying both the character set definitions of character attributes and the language-specific shift.

***error*** 16-bit unsigned integer array (required)

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Overlapping strings; <i>outbuffer</i> overwrites <i>inbuffer</i>
4	Invalid <i>bufferlength</i> parameter
5	*NLS internal error
6	*NLS internal error
7	Reserved portion of <i>flags</i> not 0
8	Upshift and downshift requested
9	Invalid table element

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

***charset*** 16-bit unsigned integer array (optional)

Passes the character set definition for the language used, as returned in NLINFO *itemnum*=12. If present, the *langnum* parameter is ignored, and this routine is much more efficient.

***shiftinfo*** character array (optional)

Passes shift information for a desired upshift or downshift (for example, as returned in NLINFO items 15 or 16 ). This parameter is utilized when bits (9:2) of *flags* are not equal to 0. If present, the *langnum* parameter is ignored, and this routine is much more efficient.

**Related Information**

Intrinsics NLCOLLATE, NLKEYCOMPARE, NLREPCHAR, NLSWITCHBUF, NLSTRANSLATE

Commands None

Manuals *Native Language Programmer's Guide* (32650-90022)

---

## NLSUBSTR

NM and CM callable.

Extracts *movelength* bytes from the *instring* to the *outstring*.

### Syntax

```
CA      I16V      CA      I16
NLSUBSTR(instring,inlength,outstring,outlength,
         I16V      16V      I16V
         startposition,movelength,langnum,
         I16V U16A  U16A
         flags,error,charset);
```

### Parameters

- instring*     **character array (required)**  
Contains the string from which the substring is extracted. The string can contain both 1 byte and 2 byte Asian characters.
- inlength*    **16-bit signed integer by value (required)**  
Contains the length, in bytes, of *instring*.
- outstring*    **character array (required)**  
Indicates where *substring* is placed.
- outlength*   **16-bit signed integer by reference (required)**  
Passes the length, in bytes, of *outstring*. After a successful call, *outlength* returns the actual length of the substring moved to *outstring*.
- startposition* **16-bit signed integer by value (required)**  
Contains the offset into *instring* where the substring starts. A value of zero is the beginning point.
- movelength*   **16-bit signed integer by value (required)**  
Passes the length, in bytes, of the substring.
- langnum*     **16-bit signed integer by value (required)**  
Contains the language ID number.



*flags***16-bit signed integer by value (required)**

Contains the flag word, and is used primarily with Asian languages. It is meaningless with 1 byte languages. It is used when special treatment is required:

<b>Bits</b>	<b>Value/Meaning</b>
12:4	Used if the first character of a substring is the second byte of a 2 byte Asian character: <ul style="list-style-type: none"> <li>0000 Return an error condition</li> <li>0001 Start from <i>startposition</i> +1</li> <li>0010 Start from <i>startposition</i> -1</li> <li>0011 Start from <i>startposition</i>, but replace character with a blank in <i>outstring</i></li> <li>0100 Start from <i>startposition</i> regardless</li> </ul>
8:4	Used if the last character in a substring is the first byte of a 2 byte Asian character: <ul style="list-style-type: none"> <li>0000 Return an error condition</li> <li>0001 Move until <i>movelength</i> +1</li> <li>0010 Move until <i>movelength</i> -1</li> <li>0011 Move until <i>movelength</i>, but replace character with a blank in <i>outstring</i></li> <li>0100 Move until <i>movelength</i> regardless</li> </ul>
0:8	Reserved for the operating system

## NLSUBSTR

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Not returned
4	Not returned
5	*NLS internal error
6	*NLS internal error
7	Invalid <i>inlength</i>
8	Invalid <i>startposition</i>
9	Invalid <i>movelength</i>
10	Reserved portion of <i>flags</i> not zero
11	Invalid value for <i>flags</i> ( 8:4)
12	Invalid value for <i>flags</i> (12:4)
13	*Start position first byte of an Asian character, or an overflow condition occurred due to <i>flags</i>
14	*End position second byte of an Asian character, or an overflow condition occurred due to <i>flags</i>

\* Do not apply to calls with *langnum* equal to 0 (NATIVE-3000).

*charset*      **16-bit unsigned integer array (optional)**

Contains the character set definition for the language to be used, as returned by *itemnum*= 12 of NLINFO. When specifying a character set, be sure that the language ID used matches the character set.

### Operation Notes

Split-stack calls are not permitted.

### Related Information

Intrinsics	NLCOLLATE, NLKEYCOMPARE, NLREPCHAR, NLSWITCHBUF, NLSTRANSLATE
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

## NLSWITCHBUF

NM and CM callable.

Converts a string of characters from phonetic order to screen order or from screen order to phonetic order.

### Syntax

```

      I16V    CA    CA
NLSWITCHBUF(langnum,instring,outstring,
      I16V    U16V    U16A
      stringlength,left-to-right,error);
```

### Parameters

- langnum***     **16-bit signed integer by value (required)**  
 Contains the language ID number.
- instring***    **character array (required)**  
 Contains the string, in phonetic order, to be converted to screen order.
- outstring***   **character array (required)**  
 Contains the string after conversion. *Outstring* and *instring* can reference the address.
- stringlength*** **16-bit signed integer by value (required)**  
 Contains the length, in bytes, of the string to be converted.
- left-to-right*** **16-bit unsigned integer by value (required)**  
 Specifies whether the implied primary mode of the data (if displayed on a terminal) is left-to-right (true) or right-to-left (false), determining which language it is and which strings of characters to switch.

## NLSWITCHBUF

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not installed
3	Invalid string length
4	Not returned
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with *langnum* equal to 0 (NATIVE3000/XL).

### Operation Notes

This intrinsic is designed to handle data for languages written from right-to-left (for example, ARABIC). Screen order is defined right-to-left if the primary mode of the terminal or printer is from right-to-left, as it is when used principally for entering or displaying data from a right-to-left language. Otherwise, screen order is defined to be left-to-right.

NLSWITCHBUF can be used by a program to convert a buffer phonetic order (the order in which the characters is typed at the terminal or spoken by a person) to screen order (the order in which the characters are displayed on a terminal screen or piece of paper). It can also convert data from screen order to phonetic order.

In general, phonetic order and screen order are not the same if USASCII text is mixed with text from a right-to-left language. The relationship between phonetic order and screen order is further complicated by the use of HINDI digits in ARABIC. (HINDI digits play a third role intermediate between ASCII characters and characters of the right-to-left language.)

This intrinsic is designed for a special purpose. Its primary value lies in its application to languages that are written from right-to-left and that may intermix left-to-right text (for example, the use of ENGLISH in ARABIC text).

NLSWITCHBUF can serve the needs of a general program, one not specifically designed for handling right-to-left data. A general program can call NLSWITCHBUF to convert data from phonetic order to screen order and back to phonetic order. One example is an editor that needs to track cursor movement on a terminal against a buffer of text in memory. If the data is not a right-to-left language, then this intrinsic returns the same text (unchanged); for all other languages, phonetic order and screen order are the same.

**Related Information**

Intrinsics	NLCOLLATE, NLKEYCOMPARE, NLREPCHAR, NLSTRANSLATE, NLSUBSTR
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## NLTRANSLATE

NM and CM callable.

Translates a string of characters from EBCDIC-to-ASCII or ASCII-to-EBCDIC using the appropriate native language table.

### Syntax

```
          I16V      CA      CA
NLTRANSLATE(transcode,inbuffer,outbuffer,
          I16V      I16V      U16A
          bufferlength,langnum,error,
          CA
          transtable);
```

### Parameters

*transcode*    **16-bit signed integer by value (required)**

Passes the translation direction:

Value	Meaning
1	EBCDIC-to-ASCII translation
2	ASCII-to-EBCDIC translation

*inbuffer*    **character array (required)**

Passes the string to be translated.

*outbuffer*    **character array (required)**

Contains the translated string. The parameters *inbuffer* and *outbuffer* can specify the same array.

*bufferlength*    **16-bit signed integer by value (required)**

Passes the number of bytes of *inbuffer* to be translated.

*langnum*    **16-bit signed integer by value (required)**

Contains the language ID number, specifying which translation tables are to be used.

*error*      **16-bit unsigned integer array (required)**

Returns two elements: the first element is the error number; the second element is reserved and always returns 0. The possible error number values are:

Value	Meaning
0	Successful
1	*NLS not installed
2	*Specified language not configured
3	Invalid <i>transcode</i> specified
4	Invalid length parameter
5	*NLS internal error
6	*NLS internal error

\* Do not apply to calls with a *langnum* equal to 0 (NATIVE3000/XL).

*transtable*      **character array (optional)**

Passes a translation table in a 256 byte array. Each byte contains the translation of the byte whose value is its index. This parameter corresponds to *itemnums* 13 and 14 of the NLINFO intrinsic. If present, the *langnum* parameter is ignored, and this routine is more efficient.

## Operation Notes

This intrinsic performs the same function as CTRANSLATE, using native language tables.

The *inbuffer* parameter is translated into *outbuffer* for length of *bufferlength*, using a translation table, and is determined as follows:

1. If *transtable* is present, a translation is made using *transtable*.
2. If *langnum* equals NATIVE3000/XL, a standard ASCII-to-EBCDIC or EBCDIC-to-ASCII translation is made.
3. The ASCII-to-EBCDIC or EBCDIC-to-ASCII translation table for the language specified is used.

## Related Information

Intrinsics	NLCOLLATE, NLKEYCOMPARE, NLREPCHAR, NLSUBSTR
Commands	None
Manuals	<i>Native Language Programmer's Guide</i> (32650-90022)

---

## OPENLOG

NM and CM callable.

Provides access to the user logging facility. User logging (LG) or system supervisor (OP) capability is required.

### Syntax

I32    CA    CA    I16    I16 OPENLOG( <i>index,logid,pass,mode,logstatus</i> );
---

### Parameters

- index*      **32-bit signed integer by reference (required)**
- Returns the user logging access. The User Logging facility uses the value of the *index* parameter to check the validity of subsequent calls to the other user logging intrinsics. The value returned is valid only for the process that made the OPENLOG call.
- logid*      **character array (required)**
- Passes the user logging identifier. Contains up to eight alphanumeric characters. Arrays < 8-characters must be terminated with a nonalphanumeric character.
- pass*      **character array (required)**
- Passes the password associated with the user logging identifier by the GETLOG command.
- mode*      **16-bit signed integer by reference (required)**
- Passes a value indicating whether the user logging facility suspends the process if it cannot complete the request for service immediately:

Value	Meaning
0	Wait
1	Nowait

If it is not possible to log the transaction and *mode* is set to nowait, *logstatus* indicates that it could not complete the request.



*logstatus*      **16-bit signed integer by reference (required)**

Returns one of the following values, indicating the success/failure of the call:

Value	Meaning
0	Successful
1	Nowait mode requested; user logging process busy
2	Parameter out of bounds
3	Request to open or write to a user logging process not running
5	Incorrect <i>mode</i> parameter passed
6	Request denied; user logging process suspended
7	Illegal capability; user logging (LG) or system supervisor (OP) capability required
8	Incorrect password passed
9	Error while writing to the user logging file
12	System out of disk space; user logging cannot proceed
13	Out of user logging entries
15	End-of-file on user logging file
16	Invalid user logging identifier

## Operation Notes

The user logging facility establishes a logging buffer (LOGBUFF) entry for every active process and increments the user count only at the first OPENLOG call. Other counters keep track of the number of times OPENLOG and CLOSELOG are called for each user; the counter is incremented for every OPENLOG and decremented for every CLOSELOG for each user. This ensures that the entry in LOGBUFF is released only if it is the last CLOSELOG call (that is, call counter = 0). The number of users and log entries are independent of the number of times the OPENLOG/CLOSELOG intrinsics are called within an application.

## Related Information

Intrinsics	None
Commands	LOG
Manuals	<i>User Logging Programmer's Guide</i> (32650-90027)

---

## PAUSE

NM and CM callable.

Suspends the calling process for a specified number of seconds.

### Syntax

32R PAUSE( <i>interval</i> );
----------------------------------

### Parameters

*interval*      **32-bit real by reference (required)**

Passes the amount of time, in seconds, that the process pauses. The value of *interval* must be positive. The maximum time allowed is approximately 2,147,484 seconds (almost 25 days).

### Operation Notes

BREAK or RESUME takes place without affecting the time countdown process.

**CNTL** Y causes the timer to countdown to restart at the beginning. It is all right if the traps are enabled because the handler returns to PAUSE (unless the handler has called the QUIT intrinsic.

### Condition Codes

CCE (2)      Request granted.  
CCG (0)      Request denied. Insufficient system table (timer request list) space.  
CCL (1)      Request denied. A negative value was specified for *interval*.

### Related Information

Intrinsics      None  
Commands      None  
Manuals      *Process Management Programmer's Guide* (32650-90023)

## PRINT

NM and CM callable.

Prints character string on job/session listing device.

### Syntax

CA      I16V      I16V PRINT( <i>message,length,controlcode</i> );
---

### Parameters

- message***      **character array (required)**  
 Passes the string of ASCII characters to \$STDLIST.
- length***      **16-bit signed integer by value (required)**  
 Returns the length of the character string to be passed in *message*. If *length* is positive, the length is in half words; if negative, the length is in bytes. If *length* exceeds the configured record length of the device, successive records are written only on terminals.
- controlcode***      **16-bit signed integer by value (required)**  
 Passes a carriage-control code.

### Operation Notes

The PRINT intrinsic is limited in its usefulness; the full capability of the file system is not available. For example, the FILE command is not allowed and some file intrinsics cannot be used, because the *filenum* parameter (obtained from the FOPEN/HPFOPEN intrinsic) is not available to PRINT intrinsic users.

### Condition Codes

- CCE (2)      Request granted.
- CCG (0)      End-of-data was encountered.
- CCL (1)      Request denied. An I/O error occurred. Further error analysis through the FCHECK intrinsic is not possible.

### Related Information

- Intrinsics      None
- Commands      None
- Manuals      *Accessing Files Programmer's Guide* (32650-90017) and *Point-to-Point Workstation I/O Reference Manual* (30000-90250)

---

## PRINTFILEINFO

NM and CM callable.

Prints a file information display on the job/session list device.

### Syntax

I16V PRINTFILEINFO( <i>filenum</i> );
--

### Parameters

*filenum*      16-bit signed integer by value (required)

Passes the file number of the file to be displayed.

### Operation Notes

Hierarchical directories and files can be opened using FOPEN or HPOPEN intrinsic. PRINTFILEINFO is used to display file information for any type of directory or file.

### Related Information

Intrinsics      None

Commands      None

Manuals      *Accessing Files Programmer's Guide* (32650-90017) and *Point-to-Point Workstation I/O Reference Manual* (30000-90250)

---

## PRINTOP

NM and CM callable.

Prints a character string on the system console.

### Syntax

CA	I16V	I16V
PRINTOP( <i>message,length,controlcode</i> );		

### Parameters

*message*      **character array (required)**

Passes the character string to be output. The string is limited to 72 characters, additional characters are ignored. Nonvideo-enhancing escape sequences are stripped out.

*length*        **16-bit signed integer by value (required)**

Passes the length of the output string to be transmitted. If *length* is positive, the length is in half words; if negative, the length is in bytes.

*controlcode*   **16-bit signed integer by value (required)**

Passes the cursor control value:

Value	Meaning
0	Moves the cursor to a new line after the message is printed
%320	Does not move the cursor after the message is printed

---

**Note**            If a null character (%000) is embedded in an array to be printed on the system console through PRINTOP, then PRINTOP prints only the portion of the array before the null character, regardless of the *length* specified.

---

### Condition Codes

CCE (2)        Request granted.

CCG (0)        Not returned.

CCL (1)        Request denied. A physical I/O error occurred. Further error analysis through the FCHECK intrinsic is not possible.

## PRINTOP

### Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

## PRINTOPREPLY

NM and CM callable.

Prints a character string on the system console and solicits a reply.

### Syntax

```

I16          CA    I16V I16V
length:=PRINTOPREPLY(message,length,zero,
                    CA    I16V
                    reply,maxlength);
```

### Functional Return

*length*      **16-bit signed integer (assigned functional return)**

Passes the length of the reply from the system operator. This length represents a half word count if *maxlength* is positive or a byte count if *maxlength* is negative.

If *maxlength* is zero, then this intrinsic does not solicit a reply.

If an error occurs, the value returned is zero.

The *length* parameter can be zero, in which case only the standard message prefix is written on the system console. If both *length* and *maxlength* are zero, then CCL (1) is returned.

### Parameters

*message*      **character array (required)**

Passes the characters that are output to the system console. The character string is limited to 50 characters, additional characters are ignored.

*length*      **16-bit signed integer by value (required)**

The length of the output string to be transmitted. If *length* is positive, the length is in half words; if negative, the length is in bytes. Do not specify a length of >50 bytes.

*zero*      **16-bit signed integer by value (required)**

Do not specify this parameter, but maintain parameter position with a value of 0.

*reply*      **character array (required)**

Returns the input characters read from the system console.

## PRINTOPREPLY

*maxlength*    16-bit signed integer by value (required)

The maximum length of the message to be read into the *reply* parameter. If the expected *length* is positive, it specifies a half word count; if negative, it specifies a byte count. Do not specify a length of >31 bytes.

### Condition Codes

CCE (2)        Request granted.

CCG (0)        Not returned.

CCL (1)        Request denied. A physical I/O error occurred. Further error analysis through the FCHECK intrinsic is not possible.

### Related Information

Intrinsics     None

Commands      None

Manuals        *Accessing Files Programmer's Guide* (32650-90017)



---

## PROCINFO

NM and CM callable.

Provides access to process information.

### Syntax

```

        I16    I16  I16V
PROCINFO(infoerror1,infoerror2,pin
        I16V  *
        [, itemnum1,item1] [...] [,itemnum6,item6]);
```

---

**Note** You can specify up to six *itemnum,item* pairs.

---

### Parameters

- error1***      **16-bit signed integer by reference (required)**  
 Returns a value indicating the success or failure of the intrinsic call. Refer to Table 4-25.
- error2***      **16-bit signed integer by reference (required)**  
 Returns a value that supplies additional error information in *error1*. Refer to Table 4-25.
- pin***            **16-bit signed integer by value (required)**  
 Passes the process identification number (PIN) of the process for which information is to be returned in the *item* parameter. A zero returns information about the calling process.  
 This parameter is not compatible with the *pin* parameter of the GETPROCINFO intrinsic.
- itemnum***       **16-bit signed integer by value (optional)**  
 Passes the item number of an information option, as defined in Table 4-26.
- item***            **type varies (optional)**  
 Returns information about the process specified by *pin*, as specified in Table 4-26.

## PROCINFO

**Table 4-25. PROCINFO Error1,Error2 Values**

<b>Error1</b>	<b>Description</b>	<b>Error2</b>
0	Successful	0
1	Insufficient capability	Index of offending <i>itemnum</i>
3	Required parameter address (other than <i>error1</i> ) out of bounds	Not used
4	Illegal array size	Array size passed to <b>PROCINFO</b>
5	Invalid <i>itemnum</i>	Index of offending <i>itemnum</i>
6	Invalid PIN; no information returned	-1
7	Unassigned PIN	-1
8	Unpaired <i>itemnum,item</i> parameters	Index of offending <i>itemnum,item</i> parameters
9	Invalid access to information	-1
10	Invalid address entered for item	Index of offending <i>itemnum,item</i> parameters
11	The process name cannot be represented using the desired syntax	Index of offending <i>itemnum,item</i> parameters

---

### **Note**

The process aborts if the *error1* parameter address is illegal.

If an error condition is detected while processing an information request, the index of the *itemnum*, where the offending option was located, is stored in *error2*.

---

Table 4-26. PROCINFO Itemnum/Item Values

Itemnum	Mnemonic	Item Description
1	I16	Process identification number (PIN) of calling process
2	I16	<p>Process identification number (PIN) of the parent of the process specified by <i>pin</i>.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>
3	I16	<p>Number of children of the process specified by <i>pin</i>.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>
4	I16	<p>Number of descendants (children, grandchildren, and so on) of the process specified by <i>pin</i>.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>
5	I16	<p>Number of generations (number of levels in the process tree substructure), including itself, of the process specified by <i>pin</i>.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>

**Table 4-26. PROCINFO Itemnum/Item Values (continued)**

Itemnum	Mnemonic	Item Description
6	I16A	<p>Process identification numbers (PINs) of all children of the process specified by <i>pin</i>.</p> <p>A variable number of PINs are returned; <i>item</i> must be a 16-bit signed integer array. The first element of the array must be set by the calling process to indicate the array size. The array size should include the array size element (for example, for four PINs, the first entry that contains the array size should be 5). PINs are stored into the array, one PIN per element, starting with the second element and continuing until the array is filled or all PINs are returned. If the array is not filled, the remaining unused elements are padded with zeros.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>
7	I16A	<p>Process identification numbers (PINs) of all descendants below children, grandchildren, and so on) of the process specified by <i>pin</i>.</p> <p>A variable number of PINs are returned; <i>item</i> must be a 16-bit signed integer array. The first element of the array must be set by the calling process to indicate the array size. The array size should include the array size element (for example, for four PINs, the first entry that contains the array size should be 5). PINs are stored into the array, one PIN per element, starting with the second element and continuing until the array is filled or all PINs are returned. If the array is not filled, the remaining unused elements are padded with zeros.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>
8	I16	<p>Priority number in the master queue of the process specified by <i>pin</i> (same as Bits (0:16) returned by <code>GETPROCINFO</code> intrinsic).</p>
9	I16	<p>State and activation information of the process specified by <i>pin</i> (same as Bits (16:16) returned by <code>GETPROCINFO</code> intrinsic).</p>

Table 4-26. PROCINFO Itemnum/Item Values (continued)

Itemnum	Mnemonic	Item Description
10	CA	<p>Program name that the process specified by <i>pin</i> is currently executing.</p> <p>The character array for the program name must be a minimum of 28 bytes long. The name is returned in the form <i>file.group.account</i>, where <i>file</i> is the local file name, <i>group</i> is the group name, and <i>account</i> is the account name of the file containing the program that the specified process is currently executing. The name is returned left-justified with the unused locations filled with blanks.</p> <p>If the calling process is executing in privileged mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>
12		<p>Returns a null-terminated HFS-syntax absolute pathname that the process specified by <i>pin</i> is currently executing.</p> <p>On input, the first four bytes of the buffer are interpreted as a signed integer specifying the maximum buffer size in bytes. The maximum buffer size does not include the four bytes used to represent the size.</p> <p>On output, the first four bytes of the buffer represent the pathname length excluding the null-terminator as a signed integer. The pathname is returned in the bytes following the pathname length. Bytes beyond the null-terminator should be considered undefined.</p> <p>If the maximum buffer length is not correct on input, variables allocated near the buffer can be overwritten, or an error occurs. If the process name is not expressed using HFS syntax, an error is returned in the first four bytes of the buffer.</p> <p>If the calling process is executing in privilege mode, requests for information are honored for any process. Otherwise, requests are honored as follows:</p> <ol style="list-style-type: none"> <li>1. Complete information is returned for children of the calling process itself.</li> <li>2. Information returned for descendants below children (grandchildren, great-grandchildren, and so on) and processes directly above the calling process is limited to <i>itemnums</i> 2 through 7, 10, and 12 only. Process handling capability is required for any user mode call unless the calling process is requesting information about itself.</li> </ol>

## PROCINFO

### Condition Codes

- CCE (2) Request granted. *error1* and *error2* return zero.
- CCG (0) Not returned.
- CCL (1) Request denied. Error codes set and returned in *error1* and *error2*.

### Related Information

- Intrinsics None
- Commands None
- Manuals *Process Management Programmer's Guide* (32650-90023)

---

## PROCTIME

NM and CM callable.

Returns the accumulated CPU time for a process.

### Syntax

```
I32  
time:=PROCTIME;
```

### Functional Return

*time*            **32-bit signed integer (assigned functional return)**

Returns the number of milliseconds that the process has been running.

### Related Information

Intrinsics        None

Commands         REPORT

Manuals           *Process Management Programmer's Guide (32650-90023)* and *Getting System Information Programmer's Guide (32650-90018)*

---

## PUTJCW

NM and CM callable.

Assigns the value of a particular job control word (JCW) in the job control word table.

### Syntax

CA	U16	I16
PUTJCW( <i>jcwname</i> , <i>jcwvalue</i> , <i>jcwstatus</i> );		

### Parameters

*jcwname*      **character array (required)**

Passes the name of the JCW. This array can contain up to 255 characters, beginning with a letter and ending with a nonalphanumeric character, such as a blank. An @ causes all executing JCWs to be set to *jcwvalue*.

*jcwvalue*      **16-bit unsigned integer by reference (required)**

Passes the value of the JCW.

*jcwstatus*      **16-bit signed integer by reference (required)**

Returns the execution status:

Value	Meaning
0	Successful
1	Error, <i>jcwname</i> greater than 255 characters
2	Error, <i>jcwname</i> does not start with an alpha character
3	Error, JCW table overflow (no JCW with this name exists in table and unable to create new entry)
4	Error, <i>jcwname</i> cannot be a valid JCW value
5	Error, cannot assign a value to a system-reserved JCW, or a read-only variable
6	Warning, standard variable reclassified as a JCW
7	Error, cannot reclassify predefined variable

### Operation Notes

If a JCW of the same name already exists in the table, only its value is entered. If no entry exists for this name, an entry is created and its value is entered.

There are three types of JCWs in the system:

- User-defined
- System-defined
- System-reserved



An attempt to assign a value to a system-reserved JCW results in an error. The system-reserved JCWs are:

- HPDATE
- HPDAY
- HPHOUR
- HPMONTH
- HPMINUTE
- HPYEAR

### **Related Information**

Intrinsics      FINDJCW, GETJCW, SETJCW

Commands      SETJCW, SHOWJCW

Manuals      *Interprocess Communication Programmer's Guide* (32650-90019)  
and *Command Interpreter Access and Variables Programmer's Guide*  
(32650-90011)

---

## QUIT

NM and CM callable.

Aborts the calling process.

### Syntax

```
I16V  
QUIT(num);
```

### Parameters

*num*            16-bit signed integer by value (required)

Passes any integer value in the range -32767 to 32768. When executed, *num* is output as part of the abort message, as follows:

```
ABORT: PIN = pin: by QUIT; PARM = num  
PROGRAM TERMINATED IN AN ERROR STATE
```

### Operation Notes

Aborts the calling process by:

- Transmitting an abort message to the list device of the calling process.
- Setting the job/session job control word (JCW) to indicate that the program terminated in an error state.
- Deleting the calling process and all of its descendants. All resources held by the deleted processes are released. Any remaining files opened by the deleted processes are closed and assigned the same disposition they had when opened.

In batch jobs, not using the CONTINUE command results in job termination.

This intrinsic affects the system job control word (JCW).

### Related Information

Intrinsics        None

Commands        None

Manuals         *Process Management Programmer's Guide (32650-90023)*

---

## QUITPROG

NM and CM callable.

Aborts the entire user process structure.

### Syntax

```
I16V  
QUITPROG(num);
```

### Parameters

*num*            16-bit signed integer by value (required)

Passes any integer value in the range 0..16383. When executed, *num* is output as part of the abort message.

### Operation Notes

Aborts the entire user process structure (all children of the job/session main process) by:

- Transmitting an abort message to the list device of the job/session main process.
- Setting the job/session job control word (JCW) to indicate that the program terminated in an error state.
- Deleting all descendants of the job/session main process (the entire user program structure). All resources held by the deleted processes are released. Any remaining files opened by the deleted processes are closed and assigned the same disposition they had when opened.

In batch jobs, not using the `CONTINUE` command terminates the job.

This intrinsic affects the system job control word (JCW).

### Related Information

Intrinsics        None

Commands        None

Manuals         *Process Management Programmer's Guide* (32650-90023)

---

## READ

NM and CM callable.

Reads an ASCII string from \$STDIN into an array.

### Syntax

I16	CA	I16V
<i>length</i> := READ( <i>message</i> , <i>msglength</i> );		

### Functional Return

*length*      **16-bit signed integer (assigned functional return)**

The length of the ASCII string that was read. If *msglength* is positive, the length specified is in half words; if negative, the length specified is in bytes.

### Parameters

*message*      **character array (required)**

Returns the ASCII characters that were read.

*msglength*    **16-bit signed integer by value (required)**

Returns the length of *message*. If *msglength* is positive, it specifies the length in half words; if negative, it specifies the length in bytes. When the record is read, the first *msglength* characters are input. If *msglength* is  $\geq$  record size of *message*, the entire record is transmitted.

---

### Note

Passing a value  $>$  *message* can cause user data corruption or a CCL (1) bounds violation if the system could be corrupted by the call.

---

### Operation Notes

Reads an ASCII string from \$STDIN into an array. This is similar to calling the FREAD intrinsic against the file \$STDIN. This intrinsic is limited; full capability of the file system is not available. For example, FILE commands are not allowed and some file intrinsics cannot be used because the *filenum* parameter, obtained from the FOPEN/HPFOPEN intrinsic, is not normally available to users of the READ intrinsic.

Basic line editing, such as cancellation of lines and backspacing, are performed automatically by the I/O driver. If the input device is a terminal in full-duplex mode with the ECHO facility on, or if the terminal is in half-duplex mode, the characters read are printed.

**READ** interprets the following to be end-of-file (EOF) markers when they begin in the first column of a record:

- For sessions, a colon (:).
- For jobs, a colon (:) or the actual EOF.

When one of these indicators are encountered, CCG (0) is returned and no transfer occurs.

If `$STDIN` is redirected, only the file's actual EOF is interpreted as the valid EOF indicator.

### **Condition Codes**

CCE (2)	Request granted.
CCG (0)	Request granted. A record with a colon in the first column, signaling the end-of-data or a hardware end-of-file, was encountered.
CCL (1)	Request denied. A physical I/O error occurred. Further error analysis through the <code>FCHECK</code> intrinsic is not possible.

### **Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

---

## READX

NM and CM callable.

Reads an ASCII string from \$STDINX into an array.

### Syntax

I16	CA	I16V
<i>length</i> :=READX( <i>message</i> , <i>msglength</i> );		

### Functional Return

*length*      **16-bit signed integer (assigned functional return)**

The length of the ASCII string that was read. If *msglength* is positive, the length specified is in half words; if negative, the length specified is in bytes.

### Parameters

*message*      **character array (required)**

Returns the ASCII characters that were read.

*msglength*    **16-bit signed integer by value (required)**

Returns the length of *message*. If *msglength* is positive, it specifies the length in half words; if negative, it specifies the length in bytes. When the record is read, the first *msglength* characters are input. If *msglength* is  $\geq$  record size of *message*, the entire record is transmitted.

---

### Note

Passing a value  $>$ *message* can cause user data corruption or a CCL (1) bounds violation if the system could be corrupted by the call.

---

## Operation Notes

The **READX** intrinsic reads an ASCII string from **\$STDINX** into an array. This is similar to calling the **FREAD** intrinsic against the file **\$STDINX**. However, the **READX** intrinsic is limited in its usefulness in that the full capability of the file system is not available to you. For example, **FILE** commands are not allowed. Also, you cannot use certain file intrinsics because the *filenum* parameter, obtained from either the **FOPEN/HPFOPEN** intrinsic, is not normally available to users of the **READX** intrinsic.

Basic line editing, such as cancellation of lines and backspacing, are performed automatically by the I/O driver. If the input device is a terminal in full-duplex mode with the **ECHO** facility on, or if the terminal is in half-duplex mode, the characters read are printed.

**READX** interprets **EOD**, **EOF**, or in a job, **EOJ**, **JOB**, or **DATA** as an end-of-file indicator.

If **\$STDIN** is redirected, only the file's actual **EOF** is interpreted as the valid **EOF** indicator.

## Condition Codes

CCE (2)	Request granted.
CCG (0)	An <b>EOD</b> , <b>EOF</b> , or in a job, <b>EOJ</b> , <b>JOB</b> , or <b>DATA</b> command was encountered.
CCL (1)	Request denied. A physical I/O error occurred. Further error analysis through the <b>FCHECK</b> intrinsic is not possible.

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Accessing Files Programmer's Guide</i> (32650-90017)

---

## RECEIVEMAIL

NM and CM callable.

Receives mail from another process. Process handling (PH) capability is required.

### Syntax

U16	I16V	UDS	U16V
<i>mailstatus</i> := RECEIVEMAIL( <i>pin</i> , <i>location</i> , <i>waitflag</i> );			

### Functional Return

*mailstatus*      **16-bit unsigned integer (assigned functional return)**

Returns one of the following values:

Value	Meaning
0	Mailbox empty and <i>waitflag</i> bit (15:1)=0.
1	No message collected, the mailbox contained outgoing mail from the receiving process.
2	Message collected successfully.
3	Error occurred because an invalid <i>pin</i> was specified or a bounds check failed.
4	Request denied, <i>waitflag</i> specified that the receiving process wait for mail if the mailbox is empty, but the other process sharing the mailbox is already suspended, waiting for mail. If both processes were suspended, neither could activate the other, and they may be deadlocked.

### Parameters

*pin*      **16-bit signed integer by value (required)**

The process sending the mail. If you specify a child process, *pin* is the process identification number (PIN) of that process. If you specify a parent process, *pin* is zero.

*location*      **user-defined structure (required)**

The buffer where the message is written.



*waitflag*      **16-bit unsigned integer by value (required)**

The action to be taken if the mailbox is empty. This is determined by the setting of bit (15:1), which has the following settings:

Value	Meaning
0	Return immediately to the calling process.
1	Wait until incoming mail is ready for collection.

### Operation Notes

The RECEIVEMAIL intrinsic lets a process collect mail transmitted to it by its parent or a child. If the mailbox for the receiving process is empty, the action taken depends on the *waitflag* parameter specified in the RECEIVEMAIL intrinsic call. If the mailbox is currently in use by another process, RECEIVEMAIL waits until the mailbox is free before accessing it.

### Condition Codes

- CCE (2)      Request granted. The mail was collected (value 2 is returned to *mailstatus*) or the mail was not collected because the mailbox contained outgoing mail from the receiving process (value 1 is returned to *mailstatus*).
- CCG (0)      Request denied. An illegal *pin* parameter (value 3 is returned to *mailstatus*).
- CCL (1)      Request denied. The *location* parameter does not allow write access on the stack address (value 3 is returned to *mailstatus*) or because both sending and receiving processes are waiting for incoming mail, causing a deadlock (value 4 is returned to *mailstatus*).

### Related Information

- Intrinsics      None
- Commands      None
- Manuals      *Interprocess Communication Programmer's Guide* (32650-90019)

---

## RESETCONTROL

NM and CM callable.

Reenables the subsystem break trap which allows a process to accept other subsystem break signals.

### Syntax

```
RESETCONTROL;
```

### Operation Notes

RESETCONTROL must be called to reenables the subsystem break trap after a user-written subsystem break trap handler, previously armed by XCONTRAP, has been invoked. It is recommended that RESETCONTROL be called from outside the user-written trap handler to eliminate recursive trap handler calls, due to multiple subsystem break signals.

### Condition Codes

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. The trap handling procedure was not invoked or was not armed by a call to XCONTRAP.

### Related Information

Intrinsics	FCONTROL, FDEVICECONTROL, XCONTRAP
Commands	None
Manuals	<i>Trap Handling Programmer's Guide</i> (32650-90026)

---

## RESETDUMP

NM and CM callable.

Disables the abort stack analysis facility. Only the current process is affected.

### Syntax

```
RESETDUMP;
```

### Condition Codes

CCE (2)	Request granted.
CCG (0)	The abort stack analysis facility was disabled prior to the RESETDUMP call and remains disabled.
CCL (1)	Not returned.

### Related Information

Intrinsics	HPRESETDUMP, HPSETDUMP, SETDUMP
Commands	RESETDUMP
Manuals	<i>MPE/iX System Debug Reference Manual (32650-90013)</i>

---

## SEARCH

NM and CM callable (differences noted below).

Searches a specially-formatted array for a specified entry or name.

### Syntax

I16	CA	I16V	CA
<i>entrynum</i> :=SEARCH( <i>buffer</i> , <i>length</i> , <i>dictionary</i> , @* <i>definition</i> );			

### Functional Return

*entrynum*      **16-bit signed integer (assigned functional return)**

Returns the entry number of the name in *dictionary* that matches *buffer*. If the name specified in *buffer* is not found, a zero is returned.

### Parameters

*buffer*          **character array (required)**

Passes the name for which the search is to be performed.

*length*          **16-bit signed integer by value (required)**

Passes the length, in characters, of the array *buffer*.

*dictionary*      **character array (required)**

Passes the name of the specially formatted array that is to be searched for *buffer*.

*definition*      (NM) **32-bit address (optional)**  
(CM) **16-bit address (optional)**

Returns the address of the command entry definition in *dictionary*.

### Operation Notes

The SEARCH intrinsic requires a specially formatted array of sequential entries. Each entry in the array is formatted in the following way:

- byte 1            An integer that specifies the length (in bytes) of the entire entry. (The length includes this byte.)
- byte 2            An integer that specifies the length (in bytes) of the name.
- bytes 3 +        A string containing the name. (This name and the name length are checked against the search string for a match.)
- bytes x +        An optional string containing a user-defined definition.

## SEARCH

The last entry in the array is a zero, specifying the end of the array as an entry of zero length. The following model creates an array to expand abbreviated input (name) to its appropriate command (definition).

```
5,1,"I","IN",
6,1,"O","OUT",
7,1,"S","SKIP",
7,1,"E","EXIT",
0;
```

In this model, the first entry specifies that the entry's length is five bytes. The name ("I") is one byte and its definition is "IN". A 1 is returned in the *entrynum* signifying that the matching name was the first entry in the array. (If no matching entry is found, a 0 is returned.) The address of the definition, if specified, is returned in *definition* of the **SEARCH** intrinsic call.

Because a search is linear, the most frequently used names should appear at the beginning of the array to ensure efficient searching.

### Related Information

Intrinsics	MYCOMMAND
Commands	None
Manuals	None

---

## SENDMAIL

NM and CM callable.

Sends mail to another process. Process handling (PH) capability is required.

### Syntax

U16	I16V	I16V	UDS	U16V
<i>mailstatus</i> := SENDMAIL( <i>pin</i> , <i>length</i> , <i>location</i> , <i>waitflag</i> );				

### Functional Return

*mailstatus*      **16-bit unsigned integer (assigned functional return)**

Returns one of the following values:

Value	Meaning
0	Mail transmitted successfully and mailbox contained no previous mail.
1	Mail was transmitted successfully and mailbox contained mail sent previously that was overwritten by the new mail, or contained previous incoming/outgoing mail that was cleared.
2	Mail not transmitted successfully; mailbox contained incoming mail to be collected by the sending process (regardless of the <i>waitflag</i> setting).
3	Error occurred; an illegal <i>pin</i> parameter was specified or a bounds check failed.
4	Illegal wait request; produces a deadlock.
5	Request denied; <i>length</i> exceeded the maximum mailbox size allowed by the system.
6	Request denied; storage resources for the mail data segment were not available.

### Parameters

*pin*      **16-bit signed integer by value (required)**

The process to receive the mail. If you specify a child process, *pin* is the process identification number (PIN) of that process. If a parent process is specified, *pin* is zero.

***length***      **16-bit signed integer by value (required)**

The length of the message, in half words, transmitted from the stack of the sending process. If 0 is specified, the mailbox is emptied of any incoming or outgoing mail.

***location***      **user-defined structure (required)**

The buffer containing the message.

***waitflag***      **16-bit unsigned integer by value (required)**

Action to be taken if the mailbox contains previously sent mail, as specified in bit (15:1):

Bits	Value/Meaning
0	Cancel (overwrite) any mail sent previously with the current mail.
1	Wait until the receiving process collects the previous mail before sending current mail.

## Operation Notes

If the receiving process mailbox contains an uncollected message from the calling process, the action taken depends on the ***waitflag*** parameter specified in the **SENDMAIL** call. If the mailbox currently is being used by another process, **SENDMAIL** waits until the mailbox is free and then sends the mail.

## Condition Codes

- CCE (2)      Request granted. The mail was sent (value 0 or 1 is returned to *mailstatus*), or the mail was not sent because the mailbox contained incoming mail to be collected by the sending process (value 2 is returned to *mailstatus*).
- CCG (0)      Request denied. An illegal ***length*** parameter (value 5 is returned to *mailstatus*), an illegal ***pin*** parameter was specified (value 3 is returned to *mailstatus*), or storage for the mail data segment was not available (value 6 is returned to *mailstatus*).
- CCL (1)      Request denied. The ***length*** or ***location*** parameter does not allow write access on the stack address (value 3 is returned to *mailstatus*), or both processes are waiting to send mail (value 4 is returned to *mailstatus*).

## Related Information

- Intrinsics      None
- Commands      None
- Manuals      *Interprocess Communication Programmer's Guide* (32650-90019)

---

## SETDUMP

NM and CM callable.

Arms a call to the system debugger from a process abort.

### Syntax

```
I16V  
SETDUMP(flags);
```

### Parameters

*flags*      16-bit unsigned integer (required)

This parameter is for backward compatibility with MPE V/E systems only. Any value passed in this parameter is ignored, but a full stack trace is always generated.

### Operation Notes

If the process aborts, Debug is called with a command string that results in a full stack trace of the CM and NM data stacks and a dump of the NM registers. This output is sent to the standard list device.

If the process that aborts is being run from a job, the process terminates after the stack trace and register dump are performed. If the process is being run from a session, after the stack trace and register dump have completed, Debug stops to accept interactive commands with I/O performed at the user terminal, provided the following conditions are met:

- The abort did not occur while in system code.
- The process entered the abort code through a native mode interrupt (typically caused by arithmetic and code-related traps).

When Debug accepts interactive input, the user is free to enter any Debug command. The user may choose to resume the process or have it terminate.

If the abort is from a stack overflow, the command list is ignored, and a stack trace is sent to the standard list device, and the process terminates. No interactive debugging is allowed.

The HPSETDUMP intrinsic is a more flexible version of this intrinsic.

### Condition Codes

- CCE (2)      Request granted.
- CCG (0)      Request denied. The abort stack analysis facility was already enabled before the SETDUMP call.
- CCL (1)      Not returned.



**Related Information**

Intrinsics      HPRESETDUMP, HPSETDUMP, RESETDUMP, STACKDUMP

Commands      SETDUMP

Manuals      *MPE/iX System Debug Reference Manual (32650-90013)*

---

## SETJCW

NM and CM callable.

Sets bits in the system job control word (JCW).

### Syntax

```
U16V
SETJCW(jcword);
```

### Parameters

*jcword*      **16-bit unsigned integer by value (required)**

The user-defined bit contents of the system JCW for interprocess communication.

Bits	Value/Meaning
------	---------------

0:1	Reserved for the operating system
-----	-----------------------------------

If (0:1)=1, the system outputs the following message when the program terminates, either normally or due to an error:

```
PROGRAM TERMINATED IN ERROR STATE
(CIERR 976)
```

In batch mode, the job is terminated unless the **CONTINUE** command is used. If JCW is set to exactly %140000 (bits (0:1)=1 and (1:1)=1 only), the **CIERR 976** message is replaced by:

```
PROGRAM ABORTED PER USER REQUEST
(CIERR 989)
```

1:15	Used for any purpose.
------	-----------------------

### Related Information

Intrinsics      FINDJCW, GETJCW, PUTJCW

Commands      SETJCW, SHOWJCW

Manuals      *Interprocess Communication Programmer's Guide* (32650-90019), *Command Interpreter Access and Variables Programmer's Guide* (32650-90011), and *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364)

---

## SORTEND

NM and CM callable.

Closes the scratch file and restores the data stack to its original state.

### Syntax

```
SORTEND;
```

### Operation Notes

Signals the beginning of the SORT program if the SORTOUTPUT intrinsic is not called.

Required if SORTINIT is called. Call SORTEND either after all calls to the output file are completed by the SORTINIT intrinsic, or after all calls to the SORTOUTPUT intrinsic are completed.

---

### Note

If you are coding a CM application, you must call SORTEND in the same routine

in which you call the SORTINIT intrinsic. This is not the case when you code an

NM application.

---

### Condition Codes

CCE (2) Request granted. No error occurred during the SORT program. The value of the *failure* parameter of the SORTINIT intrinsic is set to false.

CCG (0) Not returned.

CCL (1) Request denied. An error occurred during the SORT program. The value of the *failure* parameter of the SORTINIT intrinsic is set to true.

### Related Information

Intrinsics HPSORT, HPMERGE

Commands None

Manuals *SORT-MERGE/XL Programmer's Guide* (32650-90080)

---

## SORTERRORMESS

NM and CM callable.

Retrieves and prints a message if a fatal error occurs during the SORT program.

### Syntax

I16V    CA    I16
SORTERRORMESS( <i>errorcode</i> , <i>message</i> , <i>length</i> );

### Parameters

<i>errorcode</i>	<b>16-bit signed integer by value</b> Returns the error number from SORTINIT in the <i>errorparm</i> parameter.
<i>message</i>	<b>character array</b> Returns the text of the error message. This parameter must be at least 80 characters long.
<i>length</i>	<b>16-bit signed integer</b> Returns the length of the <i>message</i> parameter.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

## SORTINIT

NM and CM callable.

Initiates the SORT program.

### Syntax

```

          I16A      I16A      I16V
SORTINIT(inputfiles,outputfiles,outputoption,
          I16V      I32V      I16V I16A
          reclength,numrecs,numkeys,keys,
          I16A      PROC      PROC  I16A
          altseq,keycompare,errorproc,statistics,
          I16      I16          I16      I16A
          failure,errorparm,spaceallocation,charseq);

```

### Parameters

*inputfiles*      **16-bit signed integer array (optional)**

Passes the file identification numbers of the input files to be sorted. The last element of this array should be set to zero. If the files are opened with either the **NOBUF** or **MR** (multirecord) access option (*aoption*), the SORT program performs the buffering and blocking/deblocking. **\$NULL** is not a valid input file.

*outputfiles*    **16-bit signed integer array (optional)**

Passes the file identification number of the output file. The second array element must be a zero, indicating the end of the array. If the files are opened with either the **NOBUF** or **MR** (multirecord) access option (*aoption*), the SORT program performs the buffering and blocking/deblocking.

## **SORTINIT**

*outputoption*    **16-bit signed integer by value (optional)**

Passes a value that determines the format of the output records:

<b>Value</b>	<b>Meaning</b>
0	Output record is the same as the input record (default).
1	Output record is a 32-bit integer whose value is the logical (relative) record number of the record.
2	Output record contains only the key fields, concatenated together with the major keys on the left, followed by the remaining keys.
3	Output record is the logical record number followed by the key fields.

*reclength*        **16-bit signed integer by value (optional)**

Passes the maximum length of a record in characters. If you do not specify *reclength*, the record length is taken from the first file specified in the *inputfiles* parameter. If this parameter is not specified, the *inputfiles* parameter must be specified.

*numrecs*          **32-bit signed integer by value (optional)**

Passes the upper bound to the number of records to be sorted. If *numrecs* is not specified (or if all input files are not on the disk), the value of 10,000 (32-bit integer) is used. Otherwise, the value of this parameter is derived from the file label (the end of file number of the input files).

*numkeys*          **16-bit signed integer by value (optional)**

Passes the number of keys used during the comparison of records. This parameter can be  $\geq 1$ . If the *numkeys* parameter is specified, the *keys* parameter must be specified and the *keycompare* parameter cannot be specified. Together, *numkeys* and *keys* describe the way records are sorted.

*keys***16-bit integer array (optional)**

Passes information about the keys used during comparison of records. If you specify the *keys* parameter, you must specify the *numkeys* parameter and must not specify *keycompare*. Together, *keys* and *numkeys* describe the way records are merged.

The first element gives the position of the first character of the key within the record. The second element gives the number of characters in the key. Bits (0:8) of the third element give the ordering sequence of the records (0 for ascending, 1 for descending). Bits (8:8) of the third element indicate the type of data according to the following convention:

Value	Meaning
0	8-bit unsigned integer
1	Twos complement (including 16-bit and 32-bit signed integer)
2	Floating-point (including real and long)
3	Packed decimal
4	Display trailing sign
5	Packed decimal with even number of digits
6	Display leading sign
7	Display leading sign separate
8	Display trailing sign separate
9	Character (using the collating sequence of <i>charseq</i> )
11	Short floating-point decimal
12	Floating-point decimal

**Note**


---

The integrity of the *keys* array must be maintained throughout the SORT operation. Do not change it until after the SORTEND intrinsic has been called.

---

## **SORTINIT**

*altseq*      **16-bit signed integer array (optional)**

Passes an alternate collating sequence. Bits (0:8) of the first element are specified according to the following table, where the sequence comprises the columns and the data comprises the rows:

excess space above nonumber table—>

	ASCII	EBCDIC	ALTSEQ
ASCII	chr(255)	chr(2)	chr(0)
EBCDIC	chr(1)	chr(255)	undefined

Bits (8:8) of the first element specify one less than the total number of characters in the collating sequence (in this instance, chr(255) or %377).

The remaining array elements comprise the actual collating sequence responsible for the particular SORT operation.

*keycompare*      **procedure (reserved)**

Do not specify this parameter, but maintain parameter position.

*errorproc*      **procedure (reserved)**

Do not specify this parameter, but maintain parameter position.

*statistics*      **16-bit signed integer array (optional)**

Returns information on the SORT operation, as follows:

<b>Value</b>	<b>Meaning</b>
1,2	Number of records sorted (32-bit integer)
3	Number of intermediate passes
4	Space available for sorting
5,6	Number of comparisons (32-bit integer)
7,8	Number of scratch file I/Os (32-bit integer)
9,10	CPU time used in milliseconds (32-bit integer)
11,12	Elapsed time in milliseconds (32-bit integer)



<i>failure</i>	<b>16-bit signed integer (optional)</b> Returns a true value (-1) if a fatal error occurs; otherwise returns a false value (0). The value is set after each call to SORTINPUT and SORTOUTPUT. Refer to the discussion of condition codes.
<i>errorparm</i>	<b>16-bit signed integer (optional)</b> Returns the SORTLIB error number if an error occurs. Use the SORTERRORMESS intrinsic to obtain the error message text.
<i>spaceallocation</i>	<b>16-bit signed integer (reserved)</b> Do not specify this parameter, but maintain parameter position.
<i>charseq</i>	<b>16-bit signed integer array (optional)</b> Passes language information in a two-element array. Set the first element to 1. Set the second element to the language ID number of the native language whose collating sequence is to be used to sort keys of type 9 (character).

### Condition Codes

CCE (2)	Request granted. No error occurred during the SORT program. The value of the <i>failure</i> parameter of the SORTINIT intrinsic is set to false.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred during the SORT program. The value of the <i>failure</i> parameter of the SORTINIT intrinsic is set to true.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## SORTINPUT

NM and CM callable.

Provides an alternative method of specifying how records are supplied to the SORT program.

### Syntax

```
CA I16V
SORTINPUT(record,length);
```

### Parameters

<i>record</i>	<b>16-bit character array</b> Passes the next input data record.
<i>length</i>	<b>16-bit signed integer</b> Passes the length of the <i>record</i> parameter value. This value should be long enough to contain all specified keys, but not longer than the record size specified in the <i>reclen</i> parameter of the SORTINIT intrinsic.

### Operation Notes

Use this intrinsic only if the *inputfiles* parameter in a call to the SORTINIT intrinsic was not used. SORTINPUT passes the input records, one at a time, to the SORT program. A call to SORTINPUT follows a call to SORTINIT and precedes all calls to the SORTOUTPUT and SORTEND intrinsics.

---

<b>Note</b>	If you are coding a CM application, you must call SORTEND in the same routine in which you call the SORTINIT intrinsic. This is not the case when you code an NM application.
-------------	---

---

### Condition Codes

CCE (2)	Request granted. No error occurred during the SORT program.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred during the SORT program.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## SORTOUTPUT

NM and CM callable.

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

### Syntax

CA    I16 SORTOUTPUT( <i>record</i> , <i>length</i> );
---

### Parameters

<i>record</i>	<b>16-bit character array</b>  Returns the next output record. The format of this record is specified in the <i>outputoption</i> parameter of the SORTINIT intrinsic.
<i>length</i>	<b>16-bit signed integer</b>  Returns the length of the <i>record</i> parameter value. When no more records remain, the value of this parameter is set to -1.

### Operation Notes

SORTOUTPUT signals the beginning of the SORT program and receives each output record from the SORT program into an array specified by the *record* parameter. SORTOUTPUT signals the end of the input process if SORTINPUT intrinsic is called. Use SORTOUTPUT only if the *outputfiles* parameter in a call to the SORTINIT intrinsic is not specified. If SORTINPUT is called, then SORTOUTPUT is called only after SORTINPUT has passed all records. A call to SORTOUTPUT always precedes a call to the SORTEND intrinsic.

---

<b>Note</b>	If you are coding a CM application, you must call SORTEND in the same routine  in which you call the SORTINIT intrinsic. This is not the case when you code an  NM application.
-------------	---

---

### Condition Codes

CCE (2)	Request granted. No error occurred during the SORT program.
CCG (0)	Not returned.
CCL (1)	Request denied. An error occurred during the SORT program.

## **SORTOUTPUT**

### **Related Information**

Intrinsics      HPSORT, HPMERGE

Commands      None

Manuals      *SORT-MERGE/XL Programmer's Guide* (32650-90080)

---

## SORTSTAT

NM and CM callable.

Prints the SORT program statistics on \$STDLIST. Call SORTSTAT after you have called the SORTEND intrinsic.

### Syntax

```
I 16A
SORTSTAT(statistics);
```

### Parameters

*statistics*      **16-bit signed integer array**

Returns MERGE program statistics, such as the number of records, the number of intermediate passes, the space available in words, the number of compares, the number of scratch file I/O operations, the CPU time in minutes, and the elapsed time in minutes.

### Related Information

Intrinsics	HPSORT, HPMERGE
Commands	None
Manuals	<i>SORT-MERGE/XL Programmer's Guide</i> (32650-90080)

---

## **SORTTITLE**

NM and CM callable.

Prints the version number and title of the SORTLIB segment on \$STDLIST.

### **Syntax**

```
SORTTITLE;
```

### **Operation Notes**

SORTTITLE prints the date and time from the DATELINE intrinsic. SORTTITLE can be called any time after the system intrinsics are declared.

### **Related Information**

Intrinsics      HPSORT, HPMERGE

Commands        None

Manuals         *SORT-MERGE/XL Programmer's Guide* (32650-90080)

## STACKDUMP

NM and CM callable (differences noted below).

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.

### Syntax

(NM and CM)

CA	I16V	I16V	I32
<code>STACKDUMP(<i>formaldesig</i>,<i>idnumber</i>,<i>flags</i>,<i>selec</i>);</code>			

(CM: SPL language only)

CA	I16V	I16V	I32
<code>STACKDUMP'(<i>formaldesig</i>,<i>idnumber</i>,<i>flags</i>,<i>selec</i>);</code>			

### Parameters

*formaldesig*

**character array (optional)**

Passes the file name of a new output file to be opened. The name is terminated by any nonalphanumeric character except a slash (/) or a period (.). The same restrictions for the *formaldesig* parameter in the `FOPEN` intrinsic apply to this parameter.

(CM: SPL language only) If the secondary entry point, `STACKDUMP'`, is used, the first byte of *formaldesig* contains the file number that was successfully opened prior to the call to `STACKDUMP'`. In this case, the file is not closed before returning to the calling program. When a file number is passed, the record length must be between 64 bytes and 512 bytes and write access must be allowed for the file.

Default: Dump is sent to `$$STDLIST`.

*idnumber*

**16-bit signed integer by value (optional)**

If the intrinsic fails due to a file system error, the file system specific error number of the failure is returned here. Any value passed into the intrinsic through this parameter is ignored.

## STACKDUMP

- flags*            **16-bit unsigned integer by value (optional)**
- This parameter is provided for compatibility with MPE V/E. If it is present, it is ignored and has no effect.
- selec*           **32-bit integer array by reference (optional)**
- This parameter is provided for compatibility with MPE V/E. If it is present in the intrinsic call, it is ignored and has no effect.

## Condition Codes

- CCE (2)           Request granted.
- CCG (0)           Request denied. An invalid address for the location of the *formaldesig* parameter was detected.
- CCL (1)           Request denied. File system error occurred during opening, writing to, or closing the file. The file error number is returned in *idnumber*.

## Related Information

- Intrinsics        DEBUG, HPDEBUG, HPRESETDUMP, HPSETDUMP, SETDUMP
- Commands         SETDUMP
- Manuals           *MPE/iX System Debug Reference Manual (32650-90013)*



## STARTSESS

NM and CM callable.

Initiates a session on the specified terminal. Programmatic sessions (PS) capability is required.

### Syntax

```

I16V    CA    I16  I32    I16A
STARTSESS(ldev,logonstring,jsid,jsnum,jsstatus);
```

### Parameters

***ldev*** 16-bit signed integer by value (required)

The logical device number of the target terminal (the terminal on which the session is to be created). The target terminal must be a real physical device, free and available to the system, job-accepting, and hardwired. In addition, the target terminal must be configured as TYPE 16, and SUBTYPE 0 or SUBTYPE 4.

The target terminal must be set to the configured baud rate, since there is no baud rate sensing by STARTSESS.

***logonstring*** character array (required)

Passes the logon string required to initiate the session on the terminal specified by *ldev*. The first characters in the string should be the session name, if specified, or the user name if no session name is assigned.

Passwords must be supplied in *logonstring*. (There is no prompting for passwords on the target terminal.) If passwords are required but not supplied in *logonstring*, STARTSESS fails.

You must place a carriage-return character (%15) in the element following the last valid character of *logonstring*.

The *logonstring* parameter is a superset of the HELLO command syntax and includes a new option, *nowait*. The *nowait* option allows you to override the requirement of a Return key entered at the target terminal.

***jsid*** 16-bit signed integer by reference (required)

Returns the value 1, indicating that *jsnum* is a session number.

## STARTSESS

***jsnum***      **32-bit signed integer by reference (required)**  
Returns a number uniquely identifying the session that was created on the target terminal.

***jsstatus***      **16-bit signed integer array (required)**  
Returns a two-element array containing status information. The second element is reserved for the operating system. The first element returns the status information, refer to Table 4-27.

**Table 4-27. STARTSESS Status Returns**

Status Value	Meaning
0	Successful
1411	Logon failed due to IDD overflow
1412	Logon failed due to JMAT overflow
1424	Missing or invalid user name in <i>logonstring</i>
1426	Missing or invalid account name in <i>logonstring</i>
1431	Specified session user lacks interactive access (IA) capability
1436	No such group in the account
1437	No such account in the system
1438	No such user in the account
1439	User lacks a home group, and a group was not specified in <i>logonstring</i>
1444	Password required but not specified in <i>logonstring</i>
-1451	Ignored delimiter
-1452	Unknown parameter found in <i>logonstring</i> ; ignored
-1458	Bad <b>TERMTYPE</b> specified in <i>logonstring</i> ; default used
-1459	Invalid <b>PRI</b> specified in <i>logonstring</i> ; default used
-1461	<b>HIPRI</b> then <b>INPRI</b> specified; <b>INPRI</b> used
-1462	<b>INPRI</b> too low; lowest valid <b>INPRI</b> used
-1463	<b>INPRI</b> too high; highest valid <b>INPRI</b> used
-1464	<b>INPRI</b> then <b>HIPRI</b> specified; <b>HIPRI</b> used
-1465	<b>OUTCLASS</b> specified; ignored
-1473	<b>RESTART</b> specified; ignored
-1479	Invalid time specified in <i>logonstring</i> ; no time limit is used

Table 4-27. STARTSESS Status Returns (continued)

Status Value	Meaning
7000	The LDEV is out of range
7001	The LDEV must not be virtual
7002	The device specified by LDEV is not a terminal
7003	The LDEV is not free
7004	The LDEV is not job-accepting
7005	The LDEV is in diagnostic mode
7006	The LDEV terminal is down
7007	Down pending on LDEV
7008	ldev specified not a real device
7009	Caller requires programmatic sessions (PS) capability
7010	Semicolon (;) delimiter is missing
7011	The LDEV is not a valid integer
7012	The LDEV cannot be a negative number
7014	Session was aborted before logging on
7015	Logon failed because system is at its session limit
7016	Logon failed because the session's <b>INPRI</b> was $\leq$ system jobfence
7017	Logon failed because the system was unable to obtain a PCB entry for the system process
7018	Logon failed because the system was unable to obtain a DST entry for the session's data stack
7019	Logon failed because the system was unable to obtain a DST entry for the session's JIT
7020	Logon failed because the system was unable to obtain a DST entry for the session's JDT
7021	Logon failed because the system was unable to obtain a file DST entry
7022	Logon failed because the system was unable to obtain an entry for the session in the JPCNT table
7023	Logon failed because the system was unable to obtain the XDD entry for the output device
7024	Logon failed because the system was unable to obtain the XDD entry for the input device

**STARTSESS****Table 4-27. STARTSESS Status Returns (continued)**

Status Value	Meaning
7025	Unknown error occurred. This happens only in the case of an illegal call from one internal procedure to another internal procedure when passing the error number
7026	Unable to allocate <b>\$STDIN</b> , or logon failed because of a failure to open <b>\$STDIN</b>
7027	Unable to allocate <b>\$STDLIST</b> , or logon failed because of a failure to open <b>\$STDLIST</b>
7029	Logon failed because the session's account is out of CPU time
7030	Logon failed because the session's account is out of connect time
7032	Logon failed because account or user lacks IA capability
7033	Logon failed because the session's logon group is out of CPU time
7034	Logon failed because the session's logon group is out of connect time
7035	Logon string was not terminated with a carriage-return character (%15)
7036	The LDEV has an invalid type and subtype for a programmatic session
7039	Internal programmatic session error
7040	Unable to allocate <b>\$STDERR</b> , or logon failed because of a failure to open <b>\$STDERR</b>
7042	Group is not specified, and there is no default home group

---

**Note** Error messages with numbers in the 10000 range are generally Command Interpreter errors and should be looked up in the *MPE/iX Error Message Manual Volumes 1, 2 and 3* (32650-90066, 32650-90152, and 32650-90368).

---

## Operation Notes

The terminal on which the session is to be created must be available; no other user may be logged on. The target terminal is not speed-sensed, so it must be set at the configured baud rate.

When the session is created, nothing is printed to the terminal until you press the **Return** key on the selected terminal (unless you specify `nowait`). To override the need for a **Return**, you can enter the `NOWAIT` keyword parameter with the logon string when invoking the `STARTSESS` intrinsic. For example, passing the following logon string to `STARTSESS` starts a session and logon immediately without waiting for a **Return**:

```
MANAGER/password.SYS;NOWAIT
```

Never specify the `NOWAIT` option in the logon string without first ensuring that the target terminal is turned on and set at the default baud rate. If the target terminal is the system console and `nowait` is specified, then you must have system manager (SM) capability. All other valid terminals can be specified with only programmatic sessions (PS) capability.

## Related Information

Intrinsics      None

Commands      None

Manuals      *Process Management Programmer's Guide* (32650-90023)

---

## SUSPEND

NM and CM callable.

Suspends a process. Process handling (PH) capability is required.

### Syntax

```
U16V I16V
SUSPEND(allow,rin);
```

### Parameters

*allow*      **16-bit unsigned integer by value (required)**

Passes the anticipated source of the call and, later, reactivates the process, as indicated by bits (14:1) and (15:1):

Bits	Value/Meaning
15:1	Parent activation bit: 0    Process does not expect to be activated by its parent 1    Process expects to be activated by its parent
14:1	Child activation bit: 0    Process does not expect to be activated by one of its children 1    Process expects to be activated by one of its children

---

### Note

If the process is running with only process handling (PH) capability, at least one of these bits must = 1.

---

If (14:1)=1 and (15:1)=1, either the parent or a child can activate the suspended process.

Bits (0:14) are reserved for the operating system.

*rin*      **16-bit signed integer by value (optional)**

Passes the resource identification number (RIN). If *rin* is specified, it represents a local RIN that is locked by the process but is released when this process is suspended. This facility can be used to synchronize processes within the same job.

## Operation Notes

When **SUSPEND** is executed, the calling process relinquishes its access to the CPU until reactivated by an **ACTIVATE** intrinsic call. The calling process must specify the anticipated source of the **ACTIVATE** call (its parent or a child process). When the process is reactivated, it begins execution with the instruction immediately following the **SUSPEND** call.

## Condition Codes

CCE (2)	Request granted.
CCG (0)	Not returned.
CCL (1)	Request denied. The <i><b>allow</b></i> parameter is not valid, the specified RIN is not owned by this process, or the specified RIN was not locked.

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Process Management Programmer's Guide (32650-90023)</i>

---

## SWITCHDB

CM callable only.

Switches the DB register pointer. Privileged mode (PM) capability is required.

### Syntax

U16	0-P	U16V
<i>logindex</i> := SWITCHDB( <i>index</i> )		

### Functional Return

*logindex*      **16-bit unsigned integer (assigned functional return)**

Returns the logical index of the data segment indicated by the previous DB register setting, allowing this setting to be restored later. If the previous DB setting indicated the stack, a zero is returned.

### Parameters

*index*      **16-bit unsigned integer by value (required)**

Passes the logical index of the data segment where the DB register is to be switched (obtained through the GETDSEG intrinsic). The value specified is checked to ensure that the process has acquired access to the segment previously. For an extra data segment, a positive, nonzero integer must be specified. To switch to the stack segment, a zero must be specified.

### Condition Codes

CCE (2)      Request granted.  
CCG (0)      Not returned.  
CCL (1)      Request denied. An illegal data segment was specified.

### Related Information

Intrinsics      GETDSEG  
Commands      None  
Manuals      *Introduction to MPE XL for MPE V Programmers (30367-90005)*



---

## 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.

### Syntax

```
TERMINATE;
```

### Related Information

Intrinsics      None

Commands      None

Manuals      *Process Management Programmer's Guide (32650-90023)*

---

## TIMER

NM and CM callable.

Returns system timer information.

### Syntax

```
I32  
count:=TIMER;
```

### Functional Return

*count*            **32-bit signed integer (assigned functional return)**

The actual millisecond count since the midnight preceding the last system coldload.

### Operation Notes

The resolution of the system timer is one millisecond; readings taken within a one-millisecond period can be identical. The system timer is reset to zero every 24-days at 12:00 midnight. If using the `TIMER` intrinsic to compute elapsed time, correct for this condition by:

- Subtracting the current count from the previous count.
- If the result is negative, add 2,073,600,000 (number of milliseconds in 24 days).

### Related Information

Intrinsics        None

Commands        None

Manuals         *Getting System Information Programmer's Guide (32650-90018)*

---

## UNLOADPROC

NM and CM callable.

Dynamically unloads a compatibility mode (CM) segmented library (SL) procedure.

### Syntax

```
I16V  
UNLOADPROC(procid);
```

### Parameters

*procid*      16-bit signed integer by value (required)

Passes the procedure's identity number, which was obtained from the LOADPROC call.

### Condition Codes

CCE (2)      Request granted.

CCG (0)      Not returned.

CCL (1)      Request denied. An invalid *procid* was specified.

### Related Information

Intrinsics    None

Commands     None

Manuals      *Switch Programming Guide* (32650-90014)

---

## UNLOCKGLORIN

NM and CM callable.

The UNLOCKGLORIN intrinsic unlocks a global resource identification number (RIN) that was locked with the LOCKGLORIN intrinsic.

### Syntax

```
I 16V  
UNLOCKGLORIN(rinnum);
```

### Parameters

*rinnum*      **16-bit signed integer by value (required)**  
Passes the number of any RIN locked by the calling process. If *rinnum* does not specify a RIN locked by the calling process, no action is taken.

### Condition Codes

CCE (2)      Request granted.  
CCG (0)      Request denied. This RIN was not locked by this process.  
CCL (1)      Request denied. The specified RIN was not allocated.

### Related Information

Intrinsics      None  
Commands      None  
Manuals      *Resource Management Programmer's Guide* (32650-90024)

---

## UNLOCKLOCRIN

NM and CM callable.

The UNLOCKLOCRIN intrinsic unlocks a local resource identification number (RIN) that was locked by the LOCKLOCRIN intrinsic.

### Syntax

I16V  
UNLOCKLOCRIN(*rinum*);

### Parameters

*rinum*      16-bit signed integer by value (required)

Passes the locked RIN, designated by an integer from 1 to the value specified in the *rincount* parameter of the GETLOCRIN intrinsic call.

### Condition Codes

- |         |   |
|---------|---|
| CCE (2) | Request granted.  |
| CCG (0) | Request denied. The RIN specified is not locked by the calling process. |
| CCL (1) | Request denied. The specified RIN is not allocated to this process.     |

### Related Information

- |            |   |
|------------|---|
| Intrinsics | None  |
| Commands   | None  |
| Manuals    | <i>Resource Management Programmer's Guide</i> (32650-90024) |

---

## WHO

NM and CM callable.

Returns the access mode and attributes of the user calling the intrinsic.

### Syntax

U16	I32	I32	CA
WHO( <i>mode, capability, localattr, username,</i>			
CA	CA	CA	U16
<i>groupname, acctname, homename, term</i> );			

### Parameters

*mode*            **16-bit unsigned integer by reference (optional)**

Returns the current user's access mode. In this half word, the bits have the following meanings:

Bits	Value/Meaning
15:1	0    The job/session input file and job/session list file are not interactive. 1    The job/session input file and job/session list file form an interactive pair. A person responding through the input device can establish a dialog with a program that displays information on the list device.
14:1	0    The job/session input file and job/session list file are not duplicative. 1    The job/session input file and job/session list file form a duplicative pair. Images on the input device are duplicated automatically on the list device.
12:2	00   Not used. 01   The user is accessing the system through a session. 10   The user is accessing the system through a job. 11   Not used.
0:12	Reserved for the operating system

*capability***32-bit signed integer by reference (optional)**

Returns the user's file access attributes, user attributes, and capability class attributes if the appropriate bit = 1. The high-order half word indicates file access attributes and user attributes; the low-order half word indicates possession of capability-class attributes. The significance of these bits is as follows:

<b>Bits</b>	<b>Value/Meaning</b>
0:1	System manager (SM)
1:1	Account manager (AM)
2:1	Account librarian (AL)
3:1	Group librarian (GL)
4:1	Diagnostician (DI)
5:1	System supervisor (OP)
6:1	Volume set creation (CV)
7:1	Volume set usage (UV)
8:1	User logging (LG)
9:4	Reserved for the operating system
13:1	Communications system (CS)
14:1	Nonshareable devices (ND)
15:1	Save files (SF)
16:7	Reserved for the operating system
23:1	Batch access (BA)
24:1	Interactive access (IA)
25:1	Privileged mode (PM)
26:2	Reserved for the operating system
28:1	Multiple RINs (MR)
29:1	Reserved for the operating system
30:1	Extra data segments (DS)
31:1	Process handling (PH)

## WHO

<i>localattr</i>	<b>32-bit signed integer by reference (optional)</b> Returns the local attributes of the user, as defined by a user with the account manager attribute.
<i>username</i>	<b>character array (optional)</b> Returns the user's name in an 8-character array.
<i>groupname</i>	<b>character array (optional)</b> Returns the name of the user's logon group in an 8-character array.
<i>acctname</i>	<b>character array (optional)</b> Returns the name of the user's logon account in an 8-character array.
<i>homename</i>	<b>character array (optional)</b> Returns the name of the user's home group in an 8-character array. If a home group is not assigned, this array is filled with blanks.
<i>term</i>	<b>16-bit unsigned integer by reference (optional)</b> Returns the logical device number of the job/session input device. If this is a spooled (STREAM) batch job, then the logical device number is the virtual device.  A virtual device simulates a spooling device. Users cannot own the actual spooling device, so the virtual device allows users access to spooling. Each virtual device is temporarily assigned a logical device number for the duration of the input or output spooling process. Since the same logical device number can be reassigned to another virtual device with different physical characteristics, such as page length, the virtual device is not permanently configured in the system configuration. You can also obtain the logical device number of the virtual device with <b>FFILEINFO</b> .

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Getting System Information Programmer's Guide</i> (32650-90018) and <i>Point-to-Point Workstation I/O Reference Manual</i> (30000-90250).



## 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.

### Syntax

```

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

### Parameters

- index***      **32-bit signed integer by reference (required)**
- Passes your access to the user logging system. This is the value returned in the ***index*** parameter of the OPENLOG intrinsic.
- data***        **16-bit unsigned integer array (required)**
- Passes the information to be logged. A log record contains 128 half words of which 119 are available for writing data. The most efficient use of log file space is to structure arrays with lengths in multiples of 119 half words.
- length***      **16-bit signed integer by reference (required)**
- Passes the length of the data in ***data***. A positive value indicates half words; a negative value indicates bytes. If the length is greater than 119 half words or 238 bytes, the information in ***data*** is divided into two or more physical log records.
- mode***        **16-bit signed integer by reference (required)**
- Passes a value indicating whether the user logging facility will suspend the process if it cannot complete the request for service immediately. If it is not possible to log the transaction and ***mode*** is set to **nowait**, the WRITELOG intrinsic indicates through ***logstatus*** that it could not complete the request.

Value	Meaning
0	Wait
1	Nowait
2	Write the logging buffer to the disk log file (disk logging) or the disk buffer file (serial logging) at the first opportunity.

## WRITELOG

*logstatus*      **16-bit signed integer by reference (required)**

Returns one of the following values, indicating the success/failure of the call:

Value	Meaning
0	Successful
1	Nowait requested, user logging process busy
2	Parameter out of bounds
4	Incorrect <i>index</i> parameter passed
5	Incorrect <i>mode</i> parameter passed
9	Error occurred while writing
12	System out of disk space; user logging cannot proceed
14	Invalid access
15	End-of-file on user log file

### Related Information

Intrinsics	BEGINLOG, ENDLOG
Commands	None
Manuals	<i>User Logging Programmer's Guide</i> (32650-90027)

---

## XARITRAP

NM and CM callable (differences noted below).

Arms or disarms the user-written arithmetic trap handling procedure.

### Syntax

I*V   I32V   I32   I32 <b>XARITRAP(<i>mask,plabel,oldmask,oldplabel</i>) ;</b>
---

---

### Note

By default, all traps except IEEE floating-point exceptions are enabled, and the system trap handler is armed. Many floating-point operations result in an inexact result. Most compiler libraries doing floating-point operations result in an inexact trap if the IEEE inexact result trap is enabled. Therefore, enable the IEEE inexact result trap only if absolutely necessary.

---

### Parameters

*mask*      (NM) **32-bit signed integer by value (required)**  
               (CM) **16-bit signed integer by value (required)**

Determines which trap conditions, if enabled, invoke the user-written software trap handler, and which do not.

Value	Meaning
1	Corresponding trap condition is armed
0	Corresponding trap condition is disarmed

The bits and their associated arithmetic errors are:

(NM)

## XARITRAP

<b>Bits</b>	<b>Value/Meaning</b>
31:1	3000 mode floating-point divide by zero
30:1	Integer divide by zero
29:1	3000 mode floating-point underflow
28:1	3000 mode floating-point overflow
27:1	Integer overflow
26:1	3000 mode double-precision overflow
25:1	3000 mode double-precision underflow
24:1	3000 mode double-precision divide by zero
23:1	Decimal overflow
22:1	Invalid ASCII digit
21:1	Invalid decimal digit

<b>Bits</b>	<b>Value/Meaning</b>
19:2	Reserved for the operating system
18:1	Decimal divide by zero
17:1	IEEE floating-point inexact result
16:1	IEEE floating-point underflow
15:1	IEEE floating-point overflow
14:1	IEEE floating-point divide by zero
13:1	IEEE floating-point invalid operation
12:1	Range errors
11:1	Software-detected NIL pointer reference
10:1	Software-detected misaligned result of pointer arithmetic or error in conversion from long pointer to short pointer
9:1	Unimplemented condition traps
8:1	Paragraph stack overflow
7:1	3000 mode packed decimal error
1:6	Reserved for the operating system
0:1	Assertion trap

(CM)

<b>Bits</b>	<b>Value/Meaning</b>
15:1	3000 mode floating-point divide by zero
14:1	Integer divide by zero
13:1	3000 mode floating-point underflow
12:1	3000 mode floating-point overflow
11:1	Integer overflow
10:1	3000 mode double-precision overflow
9:1	3000 mode double-precision underflow
8:1	3000 mode double-precision divide by zero
7:1	Decimal overflow
6:1	Invalid ASCII digit
5:1	Invalid decimal digit
4:2	Reserved for the operating system
2:1	Decimal divide by zero

## XARITRAP

---

### Note

The following apply to the preceding trap conditions represented in the mask:

- NM supports IEEE and 3000 mode formats. Both execute in NM, but 3000 mode performs Hewlett-Packard 3000 type manipulations. Since it is possible to use both formats during program execution, there are separate bits in the mask for enabling/disabling traps of these formats.
  - Some error conditions specified here are not strictly arithmetic traps (for example, range errors, nil pointers, and paragraph stack overflow). However, many arithmetic traps are caught by reserved instructions that raise the conditional traps. For this reason, all are enabled/disabled by XARITRAP.
  - Some instructions that raise conditional traps are reserved to indicate the above trap conditions. A nonreserved instruction is one not generated by a compiler. If a nonreserved instruction causes a conditional trap, it is reported as an unimplemented condition trap. Only assembly language programmers can generate such a trap.
- 

### *plabel*

(NM) **32-bit signed integer by value (required)**  
(CM) **16-bit signed integer by value (required)**

Passes the address of the trap handling procedure. If the value is 0, the user-written arithmetic trap handler is disabled.

### *oldmask*

(NM) **32-bit signed integer by reference (required)**  
(CM) **16-bit signed integer by reference (required)**

Returns the value of the previous *mask*.

### *oldplabel*

(NM) **32-bit signed integer by reference (required)**  
(CM) **16-bit signed integer by reference (required)**

Returns the *plabel* of the process's previous arithmetic trap handler. If no *plabel* was previously configured, 0 is returned.

## Operation Notes

This trap can be armed for any combination of events; at any given time, there is only one user-written trap handler for all armed traps.

There is a difference between arming and enabling traps:

- Enabling a trap means that the occurrence of a trap condition is not ignored.
- Arming a trap is required so that, on a trap condition, a user-written routine is invoked and can take appropriate recovery actions.

The following summarizes what can occur when an arithmetic trap condition arises:

- If a trap is both enabled and armed, the user-written trap handler is invoked whenever a trap condition occurs.
- If a trap is enabled but not armed, one of two situations applies:
  - If a Pascal XL TRY statement is executed, control is passed to the RECOVER block by doing an ESCAPE.
  - If a Pascal XL TRY statement is executed, an error message is output and the process aborts.
- If a trap is disabled, irrespective of whether it is armed, the trap is ignored, and execution of the process continues without any interruption.

## Condition Codes

CCE (2)	Request granted. The desired traps are now armed.
CCG (0)	Request granted. All traps are now disarmed.
CCL (1)	Not returned.

## Related Information

Intrinsics	None
Commands	None
Manuals	<i>Trap Handling Programmer's Guide</i> (32650-90026)

---

## XCONTRAP

NM and CM callable (difference noted below).

Arms or disarms user-written subsystem break trap handling procedure.

### Syntax

I\*V      I\*  
XCONTRAP(*plabel*,*oldplabel*);

---

### Note

Any user trap handler cannot perform a GOTO out of that procedure. Performing an ESCAPE (Pascal) or completing the trap handling procedure are the only valid ways to return. The state of the process and the program results are not predictable after a non-local GOTO statement.

---

### Parameters

*plabel* (NM) **32-bit signed integer by value (required)**  
(CM) **16-bit signed integer by value (required)**

The label of the subsystem break trap handling procedure. It can be either an NM or a CM label. If the value is 0, the subsystem break trap handler is disarmed for the process.

Obtaining external labels for the NM trap handling procedure depends on the programming language. In Pascal/XL, for example, obtain the label by using the **waddress** function. Supply the name of your subsystem break trap handler as an argument to **waddress**.

To pass a CM label, set it up as follows:

1. Obtain the 16-bit external CM label of the CM subsystem break trap handler. One way to do this is by using the **LOADPROC** intrinsic.
2. Pass this 16-bit label in the following 32-bit format:

(NM)

Bits	Value/Meaning
31:1	Set to 1
30:1	Set to 0
29:1	Set to 1
16:13	Set to 0
0:16	16-bit external CM label



(CM)

Bits	Value/Meaning
0:16	16-bit external CM plabel

*oldplabel* (NM) **32-bit signed integer passed by reference (required)**  
 (CM) **16-bit signed integer passed by reference (required)**

Returns the plabel of the process's previous subsystem break trap handler. It can be either a CM or NM plabel, as described above. If no plabel was previously configured, *oldplabel* returns 0.

## Operation Notes

Call XCONTRAP to arm a user-written subsystem break trap handling procedure. This trap handler is invoked when an enabled subsystem break signal is received during an interactive session. Calling XCONTRAP from a job results in an error.

On most terminals, transmit the subsystem break signal by pressing **CTRL**Y (the default for sending a subsystem break signal). Subsystem break traps are also referred to as control-Y traps.

Only one process in a session can receive a subsystem break trap at any one time. The process that called XCONTRAP most recently receives the next subsystem break trap. Once a process has received a subsystem break trap, it cannot receive another until it calls the RESETCONTROL intrinsic to reenable the subsystem break trap. Only processes running in a session can enable subsystem break traps. The trap handler can be any procedure in the program or in the libraries to which the program is bound. The subsystem break trap handler has no parameters.

## Condition Codes

CCE (2)	Request granted. Trap enabled.
CCG (0)	Request granted. Trap disabled.
CCL (1)	Request denied. An illegal <i>plabel</i> or XCONTRAP was called from a job.

## Related Information

Intrinsics	FCONTROL, FDEVICECONTROL, RESETCONTROL
Commands	None
Manuals	<i>Trap Handling Programmer's Guide</i> (32650-90026) and <i>Point-to-Point Workstation I/O Reference</i> (30000-90250)

---

## XLIBTRAP

NM and CM callable (differences noted below).

Enables or disables a user-written software library trap handling procedure.

### Syntax

I\*V      I\*  
XLIBTRAP(*plabel*,*oldplabel*);

---

### Note

Any user trap handler cannot perform a GOTO out of that procedure. Performing an ESCAPE (Pascal) or completing the trap handling procedure are the only valid ways to return. The state of the process and the program results are not predictable after a non-local GOTO statement.

---

### Parameters

- plabel*            (NM) **32-bit signed integer by value (required)**  
                    (CM) **16-bit signed integer by value (required)**
- Passes the external-type label of the trap handling procedure. If the value is 0, the trap handler is disabled.
- oldplabel*        (NM) **32-bit signed integer by reference (required)**  
                    (CM) **16-bit signed integer by reference (required)**
- Returns the *plabel* of the process's previous software library trap handler. If no *plabel* was previously configured, 0 is returned.

### Operation Notes

When a program begins execution, the user-written software library trap handler is disabled automatically. When enabled by the XLIBTRAP intrinsic, and subsequently activated by an error, the user-written software library trap handler assumes control.

This trap handler can be enabled or disabled by calling the XLIBTRAP intrinsic.

### Condition Codes

- CCE (2)            Request granted. Trap enabled.
- CCG (0)            Request granted. Trap disabled.
- CCL (1)            Request denied. An illegal *plabel* was specified.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Trap Handling Programmer's Guide</i> (32650-90026)

---

## XSYSTRAP

NM and CM callable (differences noted below).

Enables or disables a user-written system trap handling procedure.

### Syntax

I\*V      I\*  
XSYSTRAP(*plabel*,*oldplabel*);

---

### Note

Any user trap handler cannot perform a GOTO out of that procedure. Performing an ESCAPE (Pascal) or completing the trap handling procedure are the only valid ways to return. The state of the process and the program results are not predictable after a non-local GOTO statement.

---

### Parameters

- plabel*            (NM) **32-bit signed integer by value (required)**  
                      (CM) **16-bit signed integer by value (required)**
- Passes the external-type label of the trap handling procedure. If the value is 0, the trap handler is disabled.
- oldplabel*        (NM) **32-bit signed integer by reference (required)**  
                      (CM) **16-bit signed integer by reference (required)**
- Returns the *plabel* of the process's previous software library trap handler. If no *plabel* was previously configured, 0 is returned.

### Operation Notes

When a program begins execution, the user-written system trap handler is disabled automatically. When enabled by the XSYSTRAP intrinsic, and subsequently activated by an error, the user-written system trap handler assumes control.

The trap handler can be enabled or disabled by calling the XSYSTRAP intrinsic.

### Condition Codes

- CCE (2)            Request granted. Trap enabled.
- CCG (0)            Request granted. Trap disabled.
- CCL (1)            Request denied. An illegal *plabel* was specified.

**Related Information**

Intrinsics	None
Commands	None
Manuals	<i>Trap Handling Programmer's Guide</i> (32650-90026)

---

## ZSIZE

NM and CM callable.

Alters current DB to Z area of the compatibility mode (CM) stack.

### Syntax

I16	I16V
<i>newsize</i> := ZSIZE( <i>size</i> );	

### Functional Return

*newsize*      **16-bit signed integer (assigned functional return)**  
Passes the size, in half words (CM words) actually granted.

### Parameters

*size*      **16-bit signed integer by value (required)**  
Passes the desired register value of Z, relative to DB. This is the number of half words (CM words) from DB to Z. This value must be  $\geq 0$ .

### Operation Notes

When called from programs running in NM, only the CM stack is affected. Programmatic expansion and contraction of the NM stack is not necessary.

Allows an NM program to alter the size of the current DB to Z area of the CM stack by adjusting the register offset of the Z address from the DB address (DB to Z).

Increases the Z address (CM stack expands) or decreases the Z address (CM stack contracts). If the Z to DB area size requested exceeds the maximum size permitted for the DL to Z (CM stack) area, only the maximum size allowed is granted.

All changes to the DB to Z area are made in increments or decrements of 128 half words (CM words); the size granted may differ from the size requested.

### Condition Codes

- CCE (2)      Request granted.
- CCG (0)      Request granted. The requested size exceeded the maximum limits of the DL to Z (CM stack) area. The maximum limit is granted, and the value is returned in *newsize*.
- CCL (1)      An illegal *size* parameter, less than  $(S - DB) + 64$  half words was specified. The minimum value is assigned by default.

**Related Information**

Intrinsics      None

Commands      None

Manuals      *Programmer's Skills Migration Guide (30367-90005).*





# Index

---

## 3

- 32-bit binary, convert from ASCII string number
  - DBINARY, 4-53
- 32-bit binary number, convert to ASCII string
  - DASCII, 4-50

## A

- Abort calling process
  - QUIT, 4-568
- Abort entire user process structure
  - QUITPROG, 4-569
- Abort errors, 2-12
- ABORTSESS
  - Abort specified job/session, 4-2
  - Intrinsic, 4-2
- Abort specified job/session
  - ABORTSESS, 4-2
- Abort stack analysis facility, disable
  - RESETDUMP, 4-577
- Accept another CONTROL-Y signal
  - RESETCONTROL, 4-576
- Access, close user logging facility
  - CLOSELOG, 4-32
- Access, establish for file on device
  - HPFOPEN, 4-331
- Access file, predefine characteristics
  - FOPEN, 4-189
- Access file, terminate
  - FCLOSE, 4-79
- Access information returned, file
  - FGETINFO, 4-155
- Accessing files, 1-1
- Access job and session information
  - JOBINFO, 4-452
- Access message catalogs
  - GENMESSAGE, 4-267
- Access message facility and output messages
  - CATREAD, 4-25
- Access mode, control of files/devices
  - FSETMODE, 4-247
- Access system services, 1-1
- Access to process information
  - PROCINFO, 4-559
- Access to User Logging facility
  - OPENLOG, 4-550

- acquire local RINs
  - GETLOCRIN, 4-278
- ACTIVATE
  - Activate new process, 4-4
  - Activate suspended process, 4-4
  - Intrinsic, 4-4
- Activate new process
  - ACTIVATE, 4-4
- Activate suspended process
  - ACTIVATE, 4-4
- Activation call for process, return source
  - GETORIGIN, 4-279
- Adjust directory space
  - ADJUSTUSLF, 4-6
- ADJUSTUSLF
  - Adjust directory space, 4-6
  - Intrinsic, 4-6
- Alignment, parameters, 2-2
- Allocation to job, free local RINs
  - FREELOCRIN, 4-240
- ALMANAC
  - Intrinsic, 4-8
  - Return numeric date information, 4-8
- ALTDSEG
  - Alter size of extra data segment, 4-10
  - Intrinsic, 4-10
- Alter DB to Z area of compatibility mode stack
  - ZSIZE, 4-628
- Altering files, 1-1
- Alter size of extra data segment
  - ALTDSEG, 4-10
- Application message catalog file, close
  - CATCLOSE, 4-22
- Application message catalog, open
  - CATOPEN, 4-23
- Arithmetic trap, enable/disable
  - ARITRAP, 4-12
  - HPENBLTRAP, 4-316
- Arithmetic trap handling procedure, arm/disarm
  - XARITRAP, 4-617
- ARITRAP
  - Collectively enables/disables arithmetic traps, 4-12
  - Intrinsic, 4-12
- Arm/disarm arithmetic trap handling procedure
  - XARITRAP, 4-617

- Arm/disarm software library trap handling procedure
  - XLIBTRAP, 4-624
- Arm/disarm subsystem break trap handling procedure
  - XCONTRAP, 4-622
- Arm/disarm system trap handling procedure
  - XSYSTRAP, 4-626
- Arm system debugger call from process abort
  - HPSETDUMP, 4-401
- Arm system debugger from process abort
  - SETDUMP, 4-582
- Array, search for specified entry/name
  - SEARCH, 4-578
- ASCII
  - Convert 16-bit binary to numeric ASCII string, 4-15
    - Intrinsic, 4-15
- ASCII string conversion to EBCDIC
  - CTRANSLATE, 4-48
- ASCII string, convert from 32-bit binary number
  - DASCH, 4-50
- ASCII string, convert to 16-bit binary
  - BINARY, 4-19
- ASCII string number, convert to 32-bit binary
  - DBINARY, 4-53
- ASCII string, read \$STDIN into array
  - READ, 4-570
- ASCII string, read \$STDINX into array
  - READX, 4-572
- Assign system-defined files and create process
  - CREATEPROCESS, 4-43
- Assign value of JCW in JCW table
  - PUTJCW, 4-566
- Attribute, retrieve session-level
  - HPCIGETVAR, 4-301

## B

- BEGINLOG
  - Intrinsic, 4-17
  - Mark beginning of user logging transaction, 4-17
- BINARY
  - Convert number from ASCII string to 16-bit binary, 4-19
    - Intrinsic, 4-19
- Binary conversion, 16-bit to numeric ASCII string
  - ASCII, 4-15
- Binary floating-point formats, convert data
  - HPFP\_CONVERT, 4-364
- Break mode, place session in
  - CAUSEBREAK, 4-28

## C

- CALENDAR
  - Intrinsic, 4-21
  - Return calendar date, 4-21
- Calendar date, format
  - FMTCALENDAR, 4-185
  - FMTDATE, 4-187
- Call description, intrinsic, 2-3
- Calling intrinsics, 2-1
- Calling process, abort
  - QUIT, 4-568
- Calling process, delete
  - TERMINATE, 4-607
- Calling process, set condition code
  - HPSETCCODE, 4-399
- Calling process, suspend for specified time
  - PAUSE, 4-552
- Calls
  - Errors, intrinsic, 2-8
  - Intrinsics, 2-1
- Capabilities required by intrinsics, 2-13
- Capability
  - Data segment management (DS), 2-14
  - Intrinsic, 2-13
  - Multiple resource identification number (MR), 2-14
  - Nonshareable device (ND), 2-14
  - Privileged mode (PM), 2-3, 2-15
  - Process handling (PH), 2-15
  - Programmatic sessions (PS), 2-15
  - System supervisor (OP), 2-14
  - User logging (LG), 2-14
- Capability checks, 2-14
- Catalog file, close application message
  - CATCLOSE, 4-22
- CATCLOSE
  - Close specified application message catalog file, 4-22
    - Intrinsic, 4-22
- CATOPEN
  - Intrinsic, 4-23
  - Open application message catalog, 4-23
- CATREAD
  - Access message facility and output messages, 4-25
    - Intrinsic, 4-25
- CAUSEBREAK
  - Intrinsic, 4-28
  - Place a session in break mode, 4-28
- Cause return from interrupt procedure
  - FINTEXTIT, 4-168
- Change DB register pointer
  - SWITCHDB, 4-606
- Change length of USL file

- EXPANDUSLF, 4-66
- Change priority of process
  - GETPRIORITY, 4-280
- Character Handling
  - NLFINDSTR, 4-500
  - NLJUDGE, 4-526
  - NLSUBSTR, 4-542
- Characters of string, replace nondisplayable
  - NLREPCHAR, 4-537
- Character string, print on listing device
  - PRINT, 4-553
- Character string, print on system console
  - PRINTOP, 4-555
- Character string, print on system console/reply
  - PRINTOPREPLY, 4-557
- Character strings, compare
  - NLCOLLATE, 4-489
  - NLCOLLATE2, 4-491
- Character strings, scan and move
  - NLSCANMOVE, 4-539
- Child of calling process
  - CREATE, 4-36
- Child process, delete
  - KILL, 4-459
- Child process, return PIN
  - GETPROCID, 4-283
- Child process, return status
  - GETPROCINFO, 4-284
- CLEANUSL
  - Delete inactive entries from USL file, 4-29
  - Intrinsic, 4-29
- CLOCK
  - Intrinsic, 4-31
  - Return system time, 4-31
- Close access to user logging facility
  - CLOSELOG, 4-32
- Close a file
  - FCLOSE, 4-79
- CLOSELOG
  - Close access to user logging facility, 4-32
  - Intrinsic, 4-32
- Close scratch file and restore data stack
  - SORTEND, 4-585
- Close specified application message catalog file
  - CATCLOSE, 4-22
- CM from NM, mixed-mode execution
  - HPSWITCHTOCM, 4-417
- CM plabel, get for switch to CM
  - HPLOADCMPROCEDURE, 4-371
- Code, condition, 2-8
- Coding procedures
  - Intrinsic, 1-3
- Collectively enables/disables arithmetic traps
  - ARITRAP, 4-12
- COMMAND
  - Execute command programmatically, 4-34
  - Intrinsic, 4-34
- Command, execute programmatically
  - HPCICOMMAND, 4-298
- Command image, parse user defined
  - MYCOMMAND, 4-484
- Command, programmatic execution
  - COMMAND, 4-34
- Compare language dependent character strings
  - NLCOLLATE, 4-489
  - NLCOLLATE2, 4-491
- Compare strings of different length
  - NLKEYCOMPARE, 4-528
- Compare string to pattern from NLMATCHINIT
  - NLMATCH, 4-531
- Compatibility mode SL procedure, unload
  - UNLOADPROC, 4-609
- Compatibility mode stack, alter DB to Z area
  - ZSIZE, 4-628
- Compatibility mode stack, expand/contract
  - DLSIZE, 4-56
- Compile pattern into NLMATCH form
  - NLMATCHINIT, 4-532
- Concatenate file name and language number
  - NLAPPEND, 4-488
- Condition code, 2-8
- Condition code, set for calling process
  - HPSETCCODE, 4-399
- Contract/expand compatibility mode stack
  - DLSIZE, 4-56
- Control access mode of files/devices
  - FSETMODE, 4-247
- Control device
  - HPDEVCONTROL, 4-310
- Control operation, on a file or device
  - FCONTROL, 4-84
- Control operations for a printer
  - FDEVICECONTROL, 4-108
- Control operations for a spooled device file
  - FDEVICECONTROL, 4-108
- CONTROL-Y signal, accept another
  - RESETCONTROL, 4-576
- Conversion, 16-bit binary to numeric ASCII
  - string
    - ASCII, 4-15
- Conversion and format information, return
  - NLNUMSPEC, 4-535
- Convert 16-bit binary to numeric ASCII string
  - ASCII, 4-15
- Convert 32-bit binary number to ASCII string
  - DASCII, 4-50
- Convert ASCII string number to 32-bit binary
  - DBINARY, 4-53
- Convert character string, phonetic to screen
  - NLSWITCHBUF, 4-545

- Convert character string, screen to phonetic  
NLSWITCHBUF, 4-545
- Convert data between binary floating-point  
formats  
HPFP\_CONVERT, 4-364
- Convert date from character string to numeric  
NLCONVCUSTDATE, 4-495
- Convert native language numbers  
NLCONVNUM, 4-497
- Convert number from ASCII string to 16-bit  
binary  
BINARY, 4-19
- Convert string from ASCII to EBCDIC  
CTRANS\_LATE, 4-48
- Convert string from EBCDIC to ASCII  
CTRANS\_LATE, 4-48
- Convert string from EBCDIK to JIS  
CTRANS\_LATE, 4-48
- Convert string from JIS to EBCDIK  
CTRANS\_LATE, 4-48
- Convert string to specified native format  
NLFMTNUM, 4-512
- Convert time format, character string to numeric  
NLCONVCLOCK, 4-493
- Copy data area to extra data segment  
DMOVOUT, 4-61
- Copy data from extra data segment  
DMOVIN, 4-58
- CPU time, return for process  
PROCTIME, 4-565
- CREATE
  - Create a process, 4-36
  - Intrinsic, 4-36
- Create a process  
CREATE, 4-36
- Create extra data segment  
GETDSEG, 4-272
- Create file on shareable device  
HPFOPEN, 4-331
- CREATEPROCESS
  - Create process and assign system-defined files,  
4-43
  - Intrinsic, 4-43
- Create process and assign system-defined files  
CREATEPROCESS, 4-43
- Creates pipe file type object  
HPPIPE, 4-392
- CTRANS\_LATE
  - Convert string from ASCII to EBCDIC, 4-48
  - Convert string from EBCDIC to ASCII, 4-48
  - Convert string from EBCDIK to JIS, 4-48
  - Convert string from JIS to EBCDIK, 4-48
  - Intrinsic, 4-48
- Current date and time returned  
DATELINE, 4-52

- Current record, mark  
REMOVE, 4-243
- Current VUF, return  
HP32208, 4-287

## D

- DASCII
  - Convert 32-bit binary number to ASCII string,  
4-50
  - Intrinsic, 4-50
- Data alignment, 1-3
- Data area, copy to extra data segment  
DMOVOUT, 4-61
- Data conversion, binary floating-point formats  
HPFP\_CONVERT, 4-364
- Data reference, 1-3
- Data segment, alter size  
ALTDSEG, 4-10
- Data segment, copy data from  
DMOVIN, 4-58
- Data segment, create extra  
GETDSEG, 4-272
- Data segment management (DS)
  - Capability, 2-14
- Data segment, release extra  
FREEDSEG, 4-238
- Data stack, read from direct access disk file  
FREADDIR, 4-232
- Data stack, restore and close scratch file  
SORTEND, 4-585
- Data stack, restore and end merging  
MERGEEND, 4-474
- Data type conversion, 1-3
- Data types
  - Mapping, 2-5
- Date and time, return  
DATELINE, 4-52
- Date, convert from character string to numeric  
NLCONVCUSTDATE, 4-495
- Date, format  
FMTDATE, 4-187
- Date, format calendar  
FMTCALENDAR, 4-185
- Date, format numeric as character string  
NLFMTCUSTDATE, 4-506
- Date, format through language dependent  
template  
NLFMTCALENDAR, 4-502  
NLFMTDATE, 4-508
- Date, format through long calendar format  
NLFMTLONGCAL, 4-510
- Date information returned, numeric  
ALMANAC, 4-8
- DATELINE
  - Intrinsic, 4-52

- Return current date and time, 4-52
- Date, return calendar
  - CALENDAR, 4-21
- Day, format time
  - FMTCLOCK, 4-186
- DBINARY
  - Convert ASCII string number to 32-bit binary, 4-53
  - Intrinsic, 4-53
- Deactivate a relative I/O record
  - FDELETE, 4-107
- DEBUG
  - Intrinsic, 4-55
  - Invoke debug facility, 4-55
- Debug commands, execute defined system
  - HPDEBUG, 4-308
- Debug facility, invoke
  - DEBUG, 4-55
- Declaration
  - Intrinsics, 2-1
  - System intrinsics, 2-1
- Decrease length of USL file
  - EXPANDUSLF, 4-66
- Default parameters, 1-3
- Define
  - Itemnum and item pair options, 2-4
- Define file and establish access
  - FOPEN, 4-189
- Define parameter bit setting, 2-2
- Delete calling process
  - TERMINATE, 4-607
- Delete child process
  - KILL, 4-459
- Delete entry from session-level variable table
  - HPCIDELETEVAR, 4-300
- Delete inactive entries from USL file
  - CLEANUSL, 4-29
- Description
  - Functional return, 2-7
- Determine Byte Size of Character
  - NLJUDGE, 4-526
- Determine if file pair is inactive/duplicative
  - FRELATE, 4-241
- Determine process PIN that locked local RIN
  - LOCRINOWNER, 4-466
- Device control
  - HPDEVCONTROL, 4-310
- Device, perform control operation
  - FCONTROL, 4-84
- Devices/files, control access mode
  - FSETMODE, 4-247
- Direct access disk file, read logical record
  - FREADDIR, 4-232
- Disable abort stack analysis facility
  - RESETDUMP, 4-577
- Disable arithmetic trap
  - ARITRAP, 4-12
- Disable arithmetic traps
  - HPENBLTRAP, 4-316
- Disable/enable software interrupts
  - FINTSTATE, 4-169
- Disarm/arm arithmetic trap handling procedure
  - XARITRAP, 4-617
- Disarm/arm software library trap handling procedure
  - XLIBTRAP, 4-624
- Disarm/arm subsystem break trap handling procedure
  - XCONTRAP, 4-622
- Disarm/arm system trap handling procedure
  - XSYSSTRAP, 4-626
- Disarm system debugger from process abort
  - HPRESETDUMP, 4-394
- Disk file file label information returned
  - FLABELINFO, 4-171
- disk file, move record pointer
  - FSPACE, 4-250
- disk file record, move to buffer
  - FREADSEEK, 4-236
- disk file, remove
  - FRENAME, 4-244
- disk file, set logical record pointer
  - FPOINT, 4-221
- disk file, update logical record
  - FUPDATE, 4-253
- Display system catalog error message
  - HPERRMSG, 4-321
- DLSIZE
  - Expand/Contract compatibility mode stack, 4-56
  - Intrinsic, 4-56
- DMOVIN
  - Copy data from extra data segment, 4-58
  - Intrinsic, 4-58
- DMOVOUT
  - Copy data area to extra data segment, 4-61
  - Intrinsic, 4-61
- duplicate file descriptor
  - HPFDUPLICATE, 4-327
- duplicate handle
  - HPFDUPLICATE, 4-327
- Duplicative/inactive, determine file pair
  - FRELATE, 4-241
- Dynamically enter privileged mode
  - GETPRIVMODE, 4-282
- Dynamically load native mode library procedure
  - HPGETPROCPLABEL, 4-368
- Dynamically return program to nonprivileged mode
  - GETUSERMODE, 4-286

Dynamically unload compatibility mode SL procedure  
UNLOADPROC, 4-609  
Dynamically unlock file  
FUNLOCK, 4-252

## E

EBCDIC string conversion to ASCII  
CTRANSLATE, 4-48  
EBCDIC, translate string from/to  
NLTRANSLATE, 4-548  
EBCDIK string conversion to JIS  
CTRANSLATE, 4-48  
Enable arithmetic trap  
ARITRAP, 4-12  
Enable/disable software interrupts  
FINTSTATE, 4-169  
Enables arithmetic traps  
HPENBLTRAP, 4-316  
ENDLOG  
Intrinsic, 4-64  
Mark end of user logging transaction, 4-64  
End merge and release work area  
HPMERGEEND, 4-375  
End merging and restore data stack  
MERGEEND, 4-474  
End sort operation, release work area  
HPSORTEND, 4-403  
Enter privileged mode, dynamically  
GETPRIVMODE, 4-282  
Entries, clean inactive from USL file  
CLEANUSL, 4-29  
Entry, delete from variable table  
HPCIDELETEVAR, 4-300  
Entry, search array for specified  
SEARCH, 4-578  
Error  
Abort, 2-12  
Condition codes, 2-8  
Status parameter, 2-9  
Error information for file I/O errors, return  
FCHECK, 4-69  
Error message, get/display for system catalog  
HPERRMSG, 4-321  
Error message, retrieve if fatal  
HPMERGEERRORMESS, 4-376  
HPSORTERRORMESS, 4-405  
Error, read from process stack  
HPERRREAD, 4-325  
Error stack, return depth of process  
HPERRDEPTH, 4-319  
Establish access for file on device  
HPFOPEN, 4-331  
Exception condition  
file, 4-395

Exclusive use, lock file  
FLOCK, 4-182  
Execute command programmatically  
COMMAND, 4-34  
HPCICOMMAND, 4-298  
Execute defined system debug commands  
HPDEBUG, 4-308  
Execute full stack trace  
STACKDUMP, 4-597  
Execution errors, 2-8  
Execution of NM to CM, mixed-mode  
HPSWITCHTOCM, 4-417  
Execution privileges, 1-1  
Expand/contract compatibility mode stack  
DLSIZE, 4-56  
EXPANDUSLF  
Change length of USL file, 4-66  
Intrinsic, 4-66  
Extensibility without source code changes, 1-3  
External declarations  
Data checking, 1-3  
External routines, 1-3  
Extract Bytes  
NLSUBSTR, 4-542  
Extra data segment, copy data from  
DMOVIN, 4-58  
Extra data segment, copy from data area  
DMOVOUT, 4-61  
Extra data segment, create  
GETDSEG, 4-272  
Extra data segment, release  
FREEDSEG, 4-238

## F

Facility access, close user logging  
CLOSELOG, 4-32  
Fatal error message, retrieve  
HPMERGEERRORMESS, 4-376  
HPSORTERRORMESS, 4-405  
Fatal error, retrieve and print message  
SORTERRORMESS, 4-586  
Fatal error, retrieve message  
MERGEERRORMESS, 4-475  
FATHER  
Intrinsic, 4-68  
Request PIN of parent process, 4-68  
FCHECK  
Intrinsic, 4-69  
Return error information for file I/O errors,  
4-69  
FCHECK error number returned  
FERRMSG, 4-130  
FCLOSE  
Close a file, 4-79  
Intrinsic, 4-79

**FCONTROL**  
 Intrinsic, 4-84  
 Perform control operation on a file or device,  
 4-84

**FDELETE**  
 Deactivate a relative I/O record, 4-107  
 Intrinsic, 4-107

**FDEVICECONTROL**  
 Control operations for printer or spooled  
 device file, 4-108  
 Intrinsic, 4-108

**FERRMSG**  
 Intrinsic, 4-130  
 Returns message corresponding to FCHECK  
 error number, 4-130

**FFILEINFO**  
 Intrinsic, 4-131  
 Returns information about a file, 4-131

**FFINDBYKEY**  
 Intrinsic, 4-151  
 Position record pointer, 4-151

**FFINDN**  
 Intrinsic, 4-153  
 Position record pointer, 4-153

**FGETINFO**  
 Intrinsic, 4-155  
 Return access and status information for file,  
 4-155

**FGETKEYINFO**  
 Intrinsic, 4-164  
 Request KSAM access information, 4-164  
 Request KSAM status information, 4-164  
 Status information for KSAM file, 4-164

File access  
 FOPEN, 4-189

File access, terminate  
 FCLOSE, 4-79

File, create on shareable device  
 HPFOPEN, 4-331

File designator, parse and validate  
 FPARSE, 4-218

File display, print on list device  
 PRINTFILEINFO, 4-554

File has exception condition  
 HPSELECT, 4-395

File information returned  
 FFILEINFO, 4-131

File information returned, access  
 FGETINFO, 4-155

File information returned, status  
 FGETINFO, 4-155

File I/O errors, error information returned  
 FCHECK, 4-69

File label information, disk file returned  
 FLABELINFO, 4-171

File label, read  
 FREADLABEL, 4-234

File, lock for exclusive use  
 FLOCK, 4-182

File name, concatenate  
 NLAPPEND, 4-488

File name of first library, return  
 HPFIRSTLIBRARY, 4-329

File pair, determine if inactive/duplicative  
 FRELATE, 4-241

File, perform control operation  
 FCONTROL, 4-84

File, read logical record to stack  
 FREAD, 4-223

File ready to perform read  
 HPSELECT, 4-395

File ready to perform write  
 HPSELECT, 4-395

Files/devices, control access mode  
 FSETMODE, 4-247

File, unlock dynamically  
 FUNLOCK, 4-252

**FINDJCW**  
 Intrinsic, 4-166  
 Search JCW table and return value, 4-166

**FINTEXTIT**  
 Cause return from interrupt procedure, 4-168  
 Intrinsic, 4-168

**FINTSTATE**  
 Enable/disable software interrupts, 4-169  
 Intrinsic, 4-169

**FLABELINFO**  
 Intrinsic, 4-171  
 Return information from file label, disk file,  
 4-171

Floating-point formats, convert binary  
 HPFP\_CONVERT, 4-364

**FLOCK**  
 Intrinsic, 4-182  
 Lock file for exclusive use, 4-182

Flush contents, User Logging memory buffer  
 FLUSHLOG, 4-184

**FLUSHLOG**  
 Flush contents, User Logging memory buffer,  
 4-184  
 Intrinsic, 4-184

**FMTCALENDAR**  
 Format calendar date, 4-185  
 Intrinsic, 4-185

**FMTCLOCK**  
 Format time of day, 4-186  
 Intrinsic, 4-186

**FMTDATE**  
 Format calendar date and time of day, 4-187  
 Intrinsic, 4-187

FOPEN  
 Define file and establish access, 4-189  
 Intrinsic, 4-189  
 Format  
 Intrinsic syntax, 2-3  
 Format and conversion information, return  
 NLNUMSPEC, 4-535  
 Format calendar date  
 FMTCALENDAR, 4-185  
 Format calendar date and time of day  
 FMTDATE, 4-187  
 Format date through language dependent  
 template  
 NLFMTCALENDAR, 4-502  
 Format date through long calendar format  
 NLFMTLONGCAL, 4-510  
 Format date/time through language dependent  
 template  
 NLFMTDATE, 4-508  
 Format numeric date as character string  
 NLFMTCUSTDATE, 4-506  
 Format numeric time as character string  
 NLFMTCLOCK, 4-504  
 Format time of day  
 FMTCLOCK, 4-186  
 FPARSE  
 Intrinsic, 4-218  
 Parse and validate file designator, 4-218  
 FPOINT  
 Intrinsic, 4-221  
 Set logical record pointer for disk file, 4-221  
 FREAD  
 Intrinsic, 4-223  
 Read logical record from file to stack, 4-223  
 FREADBACKWARD  
 Intrinsic, 4-226  
 Read reverse capability, 4-226  
 FREADBYKEY  
 Intrinsic, 4-228  
 Logical record, read randomly, 4-228  
 Read logical record randomly to data stack,  
 4-228  
 FREADC  
 Intrinsic, 4-230  
 Logical record, read chronologically, 4-230  
 Read logical record chronologically to data  
 stack, 4-230  
 FREADDIR  
 Intrinsic, 4-232  
 Read logical record from direct access disk  
 file, 4-232  
 FREADLABEL  
 Intrinsic, 4-234  
 Read user file label, 4-234  
 FREADSEEK  
 Intrinsic, 4-236  
 Move record from disk file to buffer, 4-236  
 FREEDSEG  
 Intrinsic, 4-238  
 Release extra data segment, 4-238  
 Free local RINs from allocation to job  
 FREELOCRIN, 4-240  
 FREELOCRIN  
 Free local RINs from allocation to job, 4-240  
 Intrinsic, 4-240  
 FRELATE  
 Determine if file pair is inactive/duplicative,  
 4-241  
 Intrinsic, 4-241  
 REMOVE  
 Current record, mark, 4-243  
 Intrinsic, 4-243  
 Mark current record, KSAM file, 4-243  
 RENAME  
 Intrinsic, 4-244  
 Remove disk file, 4-244  
 FSETMODE  
 Control access modes of files/devices, 4-247  
 Intrinsic, 4-247  
 FSPACE  
 Intrinsic, 4-250  
 Move record pointer on tape/disk file, 4-250  
 Full stack trace, execute  
 STACKDUMP, 4-597  
 Functional return description, 2-7  
 Functional returns, intrinsic, 1-1  
 FUNLOCK  
 Dynamically unlock file, 4-252  
 Intrinsic, 4-252  
 FUPDATE  
 Intrinsic, 4-253  
 Update logical record in disk file, 4-253  
 FWRITE  
 Intrinsic, 4-255  
 Write logical/physical record from stack to  
 file, 4-255  
 FWRITEDIR  
 Intrinsic, 4-263  
 Write logical record from stack to disk file,  
 4-263  
 FWRITELABEL  
 Intrinsic, 4-265  
 Write user file label, 4-265

**G**  
 GENMESSAGE  
 Access message catalogs, 4-267  
 Intrinsic, 4-267  
 Get CM label for switch to CM  
 HPLOADCMPROCEDURE, 4-371



**GETDSEG**  
 Create extra data segment, 4-272  
 Intrinsic, 4-272

**GETINFO**  
 Intrinsic, 4-275  
 Return user-supplied information, 4-275

**GETJCW**  
 Intrinsic, 4-277  
 Return system defined JCW value, 4-277

**GETLOCRIN**  
 acquire local RINs, 4-278  
 Intrinsic, 4-278

**GETORIGIN**  
 Intrinsic, 4-279  
 Return source of activation call for process, 4-279

**GETPRIORITY**  
 Change priority of process, 4-280  
 Intrinsic, 4-280

**GETPRIVMODE**  
 Dynamically enter privileged mode, 4-282  
 Intrinsic, 4-282

**GETPROCID**  
 Intrinsic, 4-283  
 Return PIN of child process, 4-283

**GETPROCINFO**  
 Intrinsic, 4-284  
 Return status for parent/child process, 4-284

Get session-level variable value/attribute  
**HPCIGETVAR**, 4-301

Get system catalog error message  
**HPERRMSG**, 4-321

**GETUSERMODE**  
 Dynamically return program to nonprivileged mode, 4-286  
 Intrinsic, 4-286

**H**

**HP32208**  
 Current VUF, return, 4-287  
 Intrinsic, 4-287  
 Return current VUF, 4-287

**HPACDINFO**  
 Intrinsic, 4-288  
 List security information from ACD, 4-288

**HPACDPUT**  
 Intrinsic, 4-294  
 Manipulate security information in ACD, 4-294

**HPCICOMMAND**  
 Execute command programmatically, 4-298  
 Intrinsic, 4-298

**HPCIDELETEVAR**  
 Intrinsic, 4-300  
 Remove entry from session-level variable table, 4-300

**HPCIGETVAR**  
 Get session-level variable value/attribute, 4-301  
 Intrinsic, 4-301

**HPCIPUTVAR**  
 Intrinsic, 4-305  
 Set value of session-level variable, 4-305

**HPDEBUG**  
 Execute defined system debug commands, 4-308  
 Intrinsic, 4-308

**HPDEVCONTROL**  
 Control device, 4-310  
 Intrinsic, 4-310

**HPDEVCREATE**  
 Creates fifo file, 4-312  
 Creates link files, 4-312  
 Creates streams file, 4-312  
 Intrinsic, 4-312

**HPENBLTRAP**  
 Enables/disables arithmetic traps, 4-316  
 Intrinsic, 4-316

**HPERRDEPTH**  
 Intrinsic, 4-319  
 Return depth of process error stack, 4-319

**HPERRMSG**  
 Get or display system catalog error message, 4-321  
 Intrinsic, 4-321

**HPERRREAD**  
 Intrinsic, 4-325  
 Read error from process stack, 4-325

**HPFDUPLICATE**  
 duplicate file descriptor, 4-327  
 duplicate handle, 4-327  
 intrinsic, 4-327  
 share data pointer, 4-327

**HPFIRSTLIBRARY**  
 Intrinsic, 4-329  
 Return file name of first library, 4-329

**HPFOPEN**  
 Create file on shareable device, 4-331  
 Establish access for file on device, 4-331  
 Intrinsic, 4-331

**HPFP\_CONVERT**  
 Convert data between binary floating-point format, 4-364  
 Intrinsic, 4-364

**HPGETPROCPLABEL**  
 Dynamically load native mode library procedure, 4-368  
 Intrinsic, 4-368

**HPLOADCMPROCEDURE**

- Get CM plabel for switch to CM, 4-371
- Intrinsic, 4-371
- HPLOADNMPROC
  - Intrinsic, 4-373
  - Return label, NM procedure, 4-373
- HPMERGEEND
  - Intrinsic, 4-375
  - Release work area and end merge, 4-375
- HPMERGEERRORMESS
  - Intrinsic, 4-376
  - Retrieve error message if fatal, 4-376
- HPMERGEINIT
  - Initialize MERGE subsystem, 4-378
  - Intrinsic, 4-378
- HPMERGEOUTPUT
  - Intrinsic, 4-383
  - Retrieve records from MERGE, 4-383
- HPMERGESTAT
  - Intrinsic, 4-384
  - Print MERGE statistics on \$STDLIST, 4-384
- HPMERGETITLE
  - Intrinsic, 4-386
  - Print version and title on \$STDLIST, 4-386
- HPMYFILE
  - Intrinsic, 4-387
  - Return program file name that called intrinsic, 4-387
- HPMYPROGRAM
  - Intrinsic, 4-389
  - Return program file name being executed, 4-389
- HPPIPE
  - Creates pipe file type object, 4-392
  - Intrinsic, 4-392
- HPRESETDUMP
  - Disarm system debugger from process abort, 4-394
  - Intrinsic, 4-394
- HPSELECT
  - file has exception condition, 4-395
  - file ready to perform read, 4-395
  - file ready to perform write, 4-395
  - Intrinsic, 4-395
- HPSETCCODE
  - Intrinsic, 4-399
  - Set condition code for calling process, 4-399
- HPSETDUMP
  - Arm system debugger call from process abort, 4-401
  - Intrinsic, 4-401
- HPSORTEND
  - Intrinsic, 4-403
  - Release SOT work area and end sort operation, 4-403
- HPSORTERRORMESS

- Intrinsic, 4-405
- Retrieve error message if fatal error, 4-405
- HPSORTINIT
  - Initialize the SORT subsystem, 4-406
  - Intrinsic, 4-406
- HPSORTINPUT
  - Intrinsic, 4-411
  - Pass records to SORT, 4-411
- HPSORTOUTPUT
  - Intrinsic, 4-412
  - Retrieve record from SORT, 4-412
- HPSORTSTAT
  - Intrinsic, 4-414
  - Print SORT statistics on \$STDLIST, 4-414
- HPSORTTITLE
  - Intrinsic, 4-416
  - Print version and title for SORT on \$STDLIST, 4-416
- HPSWITCHTOCM
  - Intrinsic, 4-417
  - Mixed-mode execution for NM to CM, 4-417
- HPSWTONMNAME
  - Intrinsic, 4-419
  - Invoke NM procedure, 4-419
- HPSWTONMPLABEL
  - Intrinsic, 4-421
  - Invoke NM procedure, 4-421
- HPUNLOADCMPROCEDURE
  - Intrinsic, 4-423
  - Unload target procedure, 4-423
- HPVOLINFO
  - Intrinsic, 4-425
  - Return volume information, 4-425

I

- Implementation
  - Mapping data types, 2-5
- Implementing intrinsics, 2-1
- Inactive/duplicative, determine file pair
  - FRELATE, 4-241
- Inactive entries, clean from USL file
  - CLEANUSL, 4-29
- Increase length of USL file
  - EXPANDUSLF, 4-66
- Information about open User Logging file
  - LOGINFO, 4-468
  - LOGSTATUS, 4-470
- Information, return user-supplied
  - GETINFO, 4-275
- Initialize MERGE subsystem
  - HPMERGEINIT, 4-378
  - MERGEINIT, 4-476
- Initialize the SORT subsystem
  - HPSORTINIT, 4-406
- Initialize USL file to empty state

INITUSLF, 4-446  
 Initiate completion for I/O request  
   IODONTWAIT, 4-448  
   IOWAIT, 4-450  
 Initiate session on specified terminal  
   STARTSESS, 4-599  
 Initiate SORT program  
   SORTINIT, 4-587  
 INITUSLF  
   Initialize USL file to empty state, 4-446  
   Intrinsic, 4-446  
 Interface  
   Intrinsic, 1-3  
 Inter-Language calls, 1-3  
 Interrupt procedure, cause return from  
   FINTEXTIT, 4-168  
 Interrupts, enable/disable software  
   FINTSTATE, 4-169  
 intrinsic  
   HPFDUPLICATE, 4-327  
 Intrinsic  
   ABORTSESS, 4-2  
   ACTIVATE, 4-4  
   ADJUSTUSLF, 4-6  
   ALMANAC, 4-8  
   ALTDSEG, 4-10  
   ARITRAP, 4-12  
   ASCII, 4-15  
   BEGINLOG, 4-17  
   BINARY, 4-19  
   CALENDAR, 4-21  
   Call errors, 2-8  
   Calling, 2-1  
   Calls, 2-1  
   Capability, 2-13  
   CATCLOSE, 4-22  
   CATOPEN, 4-23  
   CATREAD, 4-25  
   CAUSEBREAK, 4-28  
   CLEANUSL, 4-29  
   CLOCK, 4-31  
   CLOSELOG, 4-32  
   Coding procedures, 1-3  
   COMMAND, 4-34  
   CREATE, 4-36  
   CREATEPROCESS, 4-43  
   CTRANSLATE, 4-48  
   DASCH, 4-50  
   Data type conversion, 1-3  
   DATELINE, 4-52  
   DBINARY, 4-53  
   DEBUG, 4-55  
   Declarations, 2-1  
   DLSIZE, 4-56  
   DMOVIN, 4-58  
   DMOVOUT, 4-61  
   ENDLOG, 4-64  
   EXPANDUSLF, 4-66  
   FATHER, 4-68  
   FCHECK, 4-69  
   FCLOSE, 4-79  
   FCONTROL, 4-84  
   FDELETE, 4-107  
   FDEVICECONTROL, 4-108  
   FERRMSG, 4-130  
   FFILEINFO, 4-131  
   FFINDBYKEY, 4-151  
   FFINDN, 4-153  
   FGETINFO, 4-155  
   FGETKEYINFO, 4-164  
   FINDJCW, 4-166  
   FINTEXTIT, 4-168  
   FINTSTATE, 4-169  
   FLABELINFO, 4-171  
   FLOCK, 4-182  
   FLUSHLOG, 4-184  
   FMTCALENDAR, 4-185  
   FMTCLOCK, 4-186  
   FMTDATE, 4-187  
   FOPEN, 4-189  
   FPARSE, 4-218  
   FPOINT, 4-221  
   FREAD, 4-223  
   FREADBACKWARD, 4-226  
   FREADBYKEY, 4-228  
   FREADC, 4-230  
   FREADDIR, 4-232  
   FREADLABEL, 4-234  
   FREADSEEK, 4-236  
   FREEDSEG, 4-238  
   FREELOCIN, 4-240  
   FRELATE, 4-241  
   FREMOVE, 4-243  
   FRENAME, 4-244  
   FSETMODE, 4-247  
   FSPACE, 4-250  
   Functional returns, 1-1  
   FUNLOCK, 4-252  
   FUPDATE, 4-253  
   FWRITE, 4-255  
   FWRITEDIR, 4-263  
   FWRITELABEL, 4-265  
   GENMESSAGE, 4-267  
   GETDSEG, 4-272  
   GETINFO, 4-275  
   GETJCW, 4-277  
   GETLOCIN, 4-278  
   GETORIGIN, 4-279  
   GETPRIORITY, 4-280  
   GETPRIVMODE, 4-282

GETPROCID, 4-283  
 GETPROCINFO, 4-284  
 GETUSERMODE, 4-286  
 HP32208, 4-287  
 HPACDINFO, 4-288  
 HPACDPUT, 4-294  
 HPCICOMMAND, 4-298  
 HPCIDELETEVAR, 4-300  
 HPCIGETVAR, 4-301  
 HPCIPUTVAR, 4-305  
 HPDEBUG, 4-308  
 HPDEVCONTROL, 4-310  
 HPDEVCREATE, 4-312  
 HPENBLTRAP, 4-316  
 HPERRDEPTH, 4-319  
 HPERRMSG, 4-321  
 HPERRREAD, 4-325  
 HPFIRSTLIBRARY, 4-329  
 HPFOPEN, 4-331  
 HPFP\_CONVERT, 4-364  
 HPGETPROC LABEL, 4-368  
 HPLOADCMPROCEDURE, 4-371  
 HPLOADNMPROC, 4-373  
 HPMERGEEND, 4-375  
 HPMERGEERRORMESS, 4-376  
 HPMERGEINIT, 4-378  
 HPMERGEOUTPUT, 4-383  
 HPMERGEGSTAT, 4-384  
 HPMERGEITITLE, 4-386  
 HPMYFILE, 4-387  
 HPMYPROGRAM, 4-389  
 HPPIPE, 4-392  
 HPRESETDUMP, 4-394  
 HPSELECT, 4-395  
 HPSETCCODE, 4-399  
 HPSETDUMP, 4-401  
 HPSORTEND, 4-403  
 HPSORTERRORMESS, 4-405  
 HPSORTINIT, 4-406  
 HPSORTINPUT, 4-411  
 HPSORTOUTPUT, 4-412  
 HPSORTSTAT, 4-414  
 HPSORTTITLE, 4-416  
 HPSWITCHTOCM, 4-417  
 HPSWTONMNAME, 4-419  
 HPSWTONMPLABEL, 4-421  
 HPUNLOADCMPROCEDURE, 4-423  
 HPVOLINFO, 4-425  
 Implementation, 2-1  
 INITUSLF, 4-446  
 Interface, 1-3  
 IODONTWAIT, 4-448  
 IOWAIT, 4-450  
 JOBINFO, 4-452  
 KILL, 4-459

LOADPROC, 4-460  
 LOCKGLORIN, 4-462  
 LOCKLOCRIN, 4-464  
 LOCRINOWNER, 4-466  
 LOGININFO, 4-468  
 LOGSTATUS, 4-470  
 MAIL, 4-472  
 Mechanism, 1-3  
 MERGEEND, 4-474  
 MERGEERRORMESS, 4-475  
 MERGEINIT, 4-476  
 MERGEOUTPUT, 4-481  
 MERGESTAT, 4-482  
 MERGETITLE, 4-483  
 Mnemonic, 2-3  
 MYCOMMAND, 4-484  
 NLAPPEND, 4-488  
 NLCOLLATE, 4-489  
 NLCOLLATE2, 4-491  
 NLCONVCLOCK, 4-493  
 NLCONVCUSTDATE, 4-495  
 NLCONVNUM, 4-497  
 NLFINDSTR, 4-500  
 NLFMTCALENDAR, 4-502  
 NLFMTCLOCK, 4-504  
 NLFMTCUSTDATE, 4-506  
 NLFMTDATE, 4-508  
 NLFMTLONGCAL, 4-510  
 NLFMTNUM, 4-512  
 NLGETLANG, 4-516  
 NLINFO, 4-518  
 NLJUDGE, 4-526  
 NLKEYCOMPARE, 4-528  
 NLMATCH, 4-531  
 NLMATCHINIT, 4-532  
 NLNUMSPEC, 4-535  
 NLREPCHAR, 4-537  
 NLSCANMOVE, 4-539  
 NLSUBSTR, 4-542  
 NLSWITCHBUF, 4-545  
 NLTRANSLATE, 4-548  
 OPENLOG, 4-550  
 Parameters, 1-1  
 PAUSE, 4-552  
 PRINT, 4-553  
 PRINTFILEINFO, 4-554  
 PRINTOP, 4-555  
 PRINTOPREPLY, 4-557  
 Procedure body, 1-1  
 Procedure header, 1-1  
 PROCINFO, 4-559  
 PROCTIME, 4-565  
 PUTJCW, 4-566  
 QUIT, 4-568  
 QUITPROG, 4-569

- READ, 4-570
- READX, 4-572
- RECEIVEMAIL, 4-574
- RESETCONTROL, 4-576
- RESETDUMP, 4-577
- SEARCH, 4-578
- SENDMAIL, 4-580
- SETDUMP, 4-582
- SETJCW, 4-584
- SORTEND, 4-585
- SORTERRORMESS, 4-586
- SORTINIT, 4-587
- SORTINPUT, 4-592
- SORTOUTPUT, 4-593
- SORTSTAT, 4-595
- SORTTITLE, 4-596
- STACKDUMP, 4-597
- STARTSESS, 4-599
- SUSPEND, 4-604
- SWITCHDB, 4-606
- TERMINATE, 4-607
- TIMER, 4-608
- UNLOADPROC, 4-609
- UNLOCKGLORIN, 4-610
- UNLOCKLOCRIN, 4-611
- Using, 2-1
- WHO, 4-612
- WRITELOG, 4-615
- XARITRAP, 4-617
- XCONTRAP, 4-622
- XLIBTRAP, 4-624
- XSYSTRAP, 4-626
- ZSIZE, 4-628
- Intrinsic call description
  - Syntax, 2-3
- Intrinsic mechanism, 1-3
- Intrinsic trap, 2-12
- Invoke debug facility
  - DEBUG, 4-55
- Invoke intrinsics, 2-1
- Invoke NM procedure
  - HPSWTONMNAME, 4-419
  - HPSWTONMPLABEL, 4-421
- Invoke routines
  - Intrinsic mechanism, 1-3
- IODONTWAIT
  - Initiate completion for I/O request, 4-448
  - Intrinsic, 4-448
- I/O record, deactivate
  - FDELETE, 4-107
- I/O request, initiate completion for
  - IODONTWAIT, 4-448
  - IOWAIT, 4-450
- IOWAIT
  - Initiate completion for I/O request, 4-450

- Intrinsic, 4-450
- Itemnum and item pairs
  - define options, 2-4

## J

- JCW, set bits in stream
  - SETJCW, 4-584
- JCW table search, return value
  - FINDJCW, 4-166
- JCW value, assign to JCW table
  - PUTJCW, 4-566
- JCW value, return system defined
  - GETJCW, 4-277
- JIS string conversion to EBCDIK
  - CTranslate, 4-48
- JOBINFO
  - Access job and session information, 4-452
  - Intrinsic, 4-452
- Job information, access
  - JOBINFO, 4-452

## K

- KILL
  - Delete child process, 4-459
  - Intrinsic, 4-459

## L

- Label, write user file
  - FWRITELABEL, 4-265
- Language dependent character strings, compare
  - NLCOLLATE, 4-489
  - NLCOLLATE2, 4-491
- Language dependent information, return
  - NLINFO, 4-518
- Language ID, return current
  - NLGETLANG, 4-516
- Language number, concatenate
  - NLAPPEND, 4-488
- Library, return file name of first
  - HPFIRSTLIBRARY, 4-329
- List security information from ACD
  - HPACDINFO, 4-288
- Load compatibility mode SL procedure
  - LOADPROC, 4-460
- Load native mode library procedure
  - HPGETPROCPLABEL, 4-368
- LOADPROC
  - Intrinsic, 4-460
  - Load compatibility mode SL procedure, 4-460
- Local RINs, acquire
  - GETLOCRIN, 4-278
- Local RINs, free from allocation to job
  - FREELOCRIN, 4-240
- Lock file for exclusive use

- FLOCK, 4-182
- Lock global RIN
  - LOCKGLORIN, 4-462
- LOCKGLORIN
  - Intrinsic, 4-462
  - Lock global RIN, 4-462
- Lock local RIN
  - LOCKLOCRIN, 4-464
- LOCKLOCRIN
  - Intrinsic, 4-464
  - Lock local RIN, 4-464
- LOCRINOWNER
  - Determine process PIN that locked local RIN, 4-466
  - Intrinsic, 4-466
- Logical record pointer for disk file, set
  - FPOINT, 4-221
- Logical record, read chronologically
  - FREADC, 4-230
- Logical record, read from current record pointer
  - FREADBACKWARD, 4-226
- Logical record, read from file to stack
  - FREAD, 4-223
- Logical record, read randomly
  - FREADBYKEY, 4-228
- Logical record, update in disk file
  - FUPDATE, 4-253
- Logical record, write from stack to disk file
  - FWRITEDIR, 4-263
- Logical record, write from stack to file
  - FWRITE, 4-255
- LOGINFO
  - Information about open User Logging file, 4-468
  - Intrinsic, 4-468
- LOGSTATUS
  - Information about open User Logging file, 4-470
  - Intrinsic, 4-470

## M

- MAIL
  - Intrinsic, 4-472
  - Test mailbox status, 4-472
- Mailbox status, test
  - MAIL, 4-472
- Mail, send to another process
  - SENDMAIL, 4-580
- Manage processes, 1-1
- Manipulate security information in ACD
  - HPACDPUT, 4-294
- Mapping
  - Data types, 2-5
- Mark beginning of user logging transaction
  - BEGINLOG, 4-17

- Mark current record, KSAM file
  - FREMOVE, 4-243
- Mark end of user logging transaction
  - ENDLOG, 4-64
- Mechanism
  - Intrinsic, 1-3
  - Invoke routines, 1-3
- Mechanism facilities, 2-1
- Memory buffer, flush contents
  - FLUSHLOG, 4-184
- MERGEEND
  - Intrinsic, 4-474
  - Restore data stack and end merging, 4-474
- MERGEERRORMESS
  - Intrinsic, 4-475
  - Retrieve message if fatal error, 4-475
- MERGEINIT
  - Initialize MERGE subsystem, 4-476
  - Intrinsic, 4-476
- MERGEOUTPUT
  - Intrinsic, 4-481
  - Specify record output from MERGE program, 4-481
- MERGE program, specify output
  - MERGEOUTPUT, 4-481
- MERGE, retrieve records from
  - HPMERGEOUTPUT, 4-383
- MERGESTAT
  - Intrinsic, 4-482
  - Print MERGE statistics on \$STDLIST, 4-482
- MERGE statistics, print on \$STDLIST
  - MERGESTAT, 4-482
- MERGE subsystem, initialize
  - HPMERGEINIT, 4-378
  - MERGEINIT, 4-476
- MERGETITLE
  - Intrinsic, 4-483
  - Print version number and title on \$STDLIST, 4-483
- Message catalog file, close
  - CATCLOSE, 4-22
- Message catalog, open
  - CATOPEN, 4-23
- Message catalogs, access
  - GENMESSAGE, 4-267
- Message facility access and output messages
  - CATREAD, 4-25
- Message, retrieve if fatal error
  - MERGEERRORMESS, 4-475
- Message returned, corresponds to FCHECK
  - FERRMSG, 4-130
- Mixed-mode execution for NM to CM
  - HPSWITCHTOCM, 4-417
- Mnemonic
  - Intrinsic, 2-3

Mnemonic, meaning, 2-3

mode

Option Privileged, 2-3

Move and scan character strings

NLSCANMOVE, 4-539

Move record from disk file to buffer

FREADSEEK, 4-236

Move record pointer on tape/disk file

FSPACE, 4-250

Multiple resource identification number (MR)

Capability, 2-14

MYCOMMAND

Intrinsic, 4-484

Parse user defined command image, 4-484

## N

Name, search array for specified

SEARCH, 4-578

Name translation

Case-insensitive languages, 1-3

Native mode library procedure, load

HPGETPROCPLABEL, 4-368

NLAPPEND

Concatenate file name and language number,  
4-488

Intrinsic, 4-488

NLCOLLATE

Compare language dependent character strings,  
4-489

Intrinsic, 4-489

NLCOLLATE2

Compare language dependent character strings  
from a COBOL program, 4-491

Intrinsic, 4-491

NLCONVCLOCK

Convert time format, character string to  
numeric, 4-493

Intrinsic, 4-493

NLCONVCUSTDATE

Convert date from character string to numeric,  
4-495

Intrinsic, 4-495

NLCONVNUM

Convert native language numbers, 4-497

Intrinsic, 4-497

NLFINDSTR

Intrinsic, 4-500

NLFMTCALENDAR

Format date through language dependent  
template, 4-502

Intrinsic, 4-502

NLFMTCLOCK

Format numeric time as character string,  
4-504

Intrinsic, 4-504

NLFMTCUSTDATE

Format numeric date as character string,  
4-506

Intrinsic, 4-506

NLFMTDATE

Format date/time through language dependent  
template, 4-508

Intrinsic, 4-508

NLFMTLONGCAL

Format date through long calendar format,  
4-510

Intrinsic, 4-510

NLFMTNUM

Convert string to specified native format,  
4-512

Intrinsic, 4-512

NLGETLANG

Intrinsic, 4-516

Return current language ID, 4-516

NLINFO

Intrinsic, 4-518

Return language dependent information, 4-518

NLJUDGE

Intrinsic, 4-526

NLKEYCOMPARE

Compare strings of different length, 4-528

Intrinsic, 4-528

NLMATCH

Compare string to pattern from  
NLMATCHINIT, 4-531

Intrinsic, 4-531

NLMATCHINIT

Compile pattern into **NLMATCH** form, 4-532

Intrinsic, 4-532

NLNUMSPEC

Intrinsic, 4-535

Return format and conversion information,  
4-535

NLREPCHAR

Intrinsic, 4-537

Replace nondisplayable characters of string,  
4-537

NLSCANMOVE

Intrinsic, 4-539

Move and scan character strings, 4-539

NLSUBSTR

Intrinsic, 4-542

NLSWITCHBUF

Convert character string, phonetic to screen,  
4-545

Convert character string, screen to phonetic,  
4-545

Intrinsic, 4-545

NLTRANSLATE

Intrinsic, 4-548

- Translate string from/to EBCDIC, 4-548
- NM to CM, mixed-mode execution
  - HPSWITCHTOCM, 4-417
- NonPrivileged mode, return program
  - GETUSERMODE, 4-286
- Nonshareable device (ND)
  - Capability, 2-14
- Number, convert to native format
  - NLFMTNUM, 4-512
- Numbers, convert native language
  - NLCONVNUM, 4-497
- Numeric date information, returned
  - ALMANAC, 4-8

## O

- Open application message catalog
  - CATOPEN, 4-23
- OPENLOG
  - Access to User Logging facility, 4-550
  - Intrinsic, 4-550
- Open User Logging file, information
  - LOGINFO, 4-468
  - LOGSTATUS, 4-470
- Option
  - Itemnum and item pairs, 2-4
  - Using parameter bit setting, 2-2
- Option privileged, 2-3
- Output messages and access message facility
  - CATREAD, 4-25

## P

- Parameter
  - Status error, 2-9
- Parameter alignment, 2-2
- Parameter bit setting, 2-2
- Parameter checking, 1-3
- Parameters, intrinsic, 1-1
- Parent process, request PIN of
  - FATHER, 4-68
- Parent process, return status
  - GETPROCINFO, 4-284
- Parse and validate file designator
  - FPARSE, 4-218
- Parse user defined command image
  - MYCOMMAND, 4-484
- Pass records to SORT
  - HPSORTINPUT, 4-411
- PAUSE
  - Intrinsic, 4-552
  - Suspend calling process for time specified, 4-552
- Perform control operation on a file or device
  - FCONTROL, 4-84
- Phonetic to screen, convert character string
  - NLSWITCHBUF, 4-545

- Physical record, write from stack to file
  - FWRITE, 4-255
- PIN, return for child process
  - GETPROCID, 4-283
- Place a session in break mode
  - CAUSEBREAK, 4-28
- Position record pointer
  - FFINDBYKEY, 4-151
  - FFINDN, 4-153
- Predefine characteristics of file and access
  - FOPEN, 4-189
- PRINT
  - Intrinsic, 4-553
  - Print character string on listing device, 4-553
  - Print character sting on system console
    - PRINTOP, 4-555
  - Print character string on listing device
    - PRINT, 4-553
  - Print character string on system console/reply
    - PRINTOPREPLY, 4-557
- Printer, control operations
  - FDEVICECONTROL, 4-108
- Print file display on list device
  - PRINTFILEINFO, 4-554
- PRINTFILEINFO
  - Intrinsic, 4-554
  - Print file display on list device, 4-554
- Print MERGE statistics on \$STDLIST
  - HPMERGESTAT, 4-384
  - MERGESTAT, 4-482
- PRINTOP
  - Intrinsic, 4-555
  - Print character string on system console, 4-555
- PRINTOPREPLY
  - Intrinsic, 4-557
  - Print character string on system console/reply, 4-557
- Print SORT program statistics on \$STDLIST
  - SORTSTAT, 4-595
- Print SORT statistics on \$STDLIST
  - HPSORTSTAT, 4-414
- Print version and title for SORT on \$STDLIST
  - HPSORTTITLE, 4-416
- Print version and title on \$STDLIST
  - HPMERGETITLE, 4-386
- Print version number and title on \$STDLIST
  - MERGETITLE, 4-483
- Print version number/title on \$STDLIST
  - SORTTITLE, 4-596
- Priority of process, change
  - GETPRIORITY, 4-280
- Privileged mode (PM)
  - Capability, 2-3, 2-15
- Privileged mode (PM), enter dynamically



- GETPRIVMODE, 4-282
- Procedure, arm/disarm system trap handling
  - XSYSTRAP, 4-626
- Procedure, arm/disarm trap handling
  - XARITRAP, 4-617
  - XCONTRAP, 4-622
  - XLIBTRAP, 4-624
- Procedure body
  - Intrinsic, 1-1
- Procedure header
  - Intrinsic, 1-1
- Procedure, invoke NM
  - HPSWTNMNAME, 4-419
  - HPSWTNMPLABEL, 4-421
- Procedure, unload
  - HPUNLOADCMPROCEDURE, 4-423
- Process abort, arm system debugger
  - SETDUMP, 4-582
- Process abort, arm system debugger call from
  - HPSETDUMP, 4-401
- Process abort, disarm system debugger
  - HPRESETDUMP, 4-394
- Process, accept another CONTROL-Y signal
  - RESETCONTROL, 4-576
- Process calling, abort
  - QUIT, 4-568
- Process, change priority
  - GETPRIORITY, 4-280
- Process creation
  - CREATE, 4-36
  - CREATEPROCESS, 4-43
- Process, delete calling
  - TERMINATE, 4-607
- Process error stack, return depth
  - HPERRDEPTH, 4-319
- Processes, manage, 1-1
- Process handling (PH)
  - Capability, 2-15
- Process information, provide access
  - PROCINFO, 4-559
- Process, receive mail from another
  - RECEIVEMAIL, 4-574
- Process, request PIN of parent
  - FATHER, 4-68
- Process, return CPU time
  - PROCTIME, 4-565
- Process, return source of activation
  - GETORIGIN, 4-279
- Process, send mail to another
  - SENDMAIL, 4-580
- Process stack, read error
  - HPERRREAD, 4-325
- Process structure, abort entire user
  - QUITPROG, 4-569
- Process, suspend

- SUSPEND, 4-604
- Process, suspend for specified time
  - PAUSE, 4-552
- PROCINFO
  - Access to process information, 4-559
  - Intrinsic, 4-559
- PROCTIME
  - Intrinsic, 4-565
  - Return CPU time for process, 4-565
- Program execution, capability check, 2-14
- Programmatic execution of command
  - COMMAND, 4-34
  - HPCICOMMAND, 4-298
- Programmatic sessions (PS)
  - Capability, 2-15
- Programming languages
  - Mapping, 2-5
- Program, return to nonprivileged mode
  - GETUSERMODE, 4-286
- PUTJCW
  - Assign value of JCW in JCW table, 4-566
  - Intrinsic, 4-566

**Q**

- QUIT
  - Abort calling process, 4-568
  - Intrinsic, 4-568
- QUITPROG
  - Abort entire user process structure, 4-569
  - Intrinsic, 4-569

**R**

- READ
  - Intrinsic, 4-570
  - Read \$STDIN ASCII string into array, 4-570
- Read error from process stack
  - HPERRREAD, 4-325
- Read file
  - ready, 4-395
- Read logical record chronologically to data stack
  - FREADC, 4-230
- Read logical record from current record pointer
  - FREADBACKWARD, 4-226
- Read logical record from direct access disk file
  - FREADDIR, 4-232
- Read logical record from file to stack
  - FREAD, 4-223
- Read logical record randomly to data stack
  - FREADBYKEY, 4-228
- Read reverse capability
  - FREADBACKWARD, 4-226
- Read \$STDIN ASCII string into array
  - READ, 4-570
- Read \$STDINX ASCII string into array
  - READX, 4-572

Read user file label  
     FREADLABEL, 4-234  
 READX  
     Intrinsic, 4-572  
     Read \$STDINX ASCII string into array,  
         4-572  
 RECEIVEMAIL  
     Intrinsic, 4-574  
     Receive mail from another process, 4-574  
 Receive mail from another process  
     RECEIVEMAIL, 4-574  
 Record, move from disk file to buffer  
     FREADSEEK, 4-236  
 Record output form MERGE program, specify  
     MERGEOUTPUT, 4-481  
 Record pointer, move on tape/disk file  
     FSPACE, 4-250  
 Record, retrieve from SORT  
     HPSORTOUTPUT, 4-412  
 Records, how output from SORT program  
     SORTOUTPUT, 4-593  
 Records, how supplied to SORT program  
     SORTINPUT, 4-592  
 Records, pass to SORT  
     HPSORTINPUT, 4-411  
 Records, retrieve from MERGE  
     HPMERGEOUTPUT, 4-383  
 Record, write to User Logging  
     WRITELOG, 4-615  
 Relative I/O record, deactivate  
     FDELETE, 4-107  
 Release extra data segment  
     FREEDSEG, 4-238  
 Release SORT work area and end sort operation  
     HPSORTEND, 4-403  
 Release work area and end merge  
     HPMERGEEND, 4-375  
 Remove disk file  
     FRENAME, 4-244  
 Replace nondisplayable characters of string  
     NLREPCCHAR, 4-537  
 Request KSAM access information  
     FGETKEYINFO, 4-164  
 Request KSAM status information  
     FGETKEYINFO, 4-164  
 Request PIN of parent process  
     FATHER, 4-68  
 Request utility functions, 1-1  
 Required capabilities for intrinsic use, 2-13  
 RESETCONTROL  
     Accept another CONTROL-Y signal, 4-576  
     Intrinsic, 4-576  
 RESETDUMP  
     Disable abort stack analysis facility, 4-577  
     Intrinsic, 4-577  
 Resources, system access, 1-1  
 Restore data stack and close scratch file  
     SORTEND, 4-585  
 Restore data stack and end merging  
     MERGEEND, 4-474  
 Retrieve and print message if fatal error  
     SORTERRORMESS, 4-586  
 Retrieve error message if fatal  
     HPMERGEERRORMESS, 4-376  
 Retrieve error message if fatal error  
     HPSORTERRORMESS, 4-405  
 Retrieve message if fatal error  
     MERGEERRORMESS, 4-475  
 Retrieve record from SORT  
     HPSORTOUTPUT, 4-412  
 Retrieve records from MERGE  
     HPMERGEOUTPUT, 4-383  
 Return access and status information for file  
     FGETINFO, 4-155  
 Return calendar date  
     CALENDAR, 4-21  
 Return CPU time for process  
     PROCTIME, 4-565  
 Return current date and time  
     DATELINE, 4-52  
 Return current language ID  
     NLGETLANG, 4-516  
 Return current VUF  
     HP32208, 4-287  
 Return depth of process error stack  
     HPERRDEPTH, 4-319  
 Return error information for file I/O errors  
     FCHECK, 4-69  
 Return file label information of disk file  
     FLABELINFO, 4-171  
 Return file name of first library  
     HPFIRSTLIBRARY, 4-329  
 Return format and conversion information  
     NLNUMSPEC, 4-535  
 Return from interrupt procedure  
     FINTEXTIT, 4-168  
 Return, Functional, 2-7  
 Return information about user  
     WHO, 4-612  
 Return language dependent information  
     NLINFO, 4-518  
 Return numeric date information  
     ALMANAC, 4-8  
 Return PIN of child process  
     GETPROCID, 4-283  
 Return plabel, NM procedure  
     HPLOADNMPROC, 4-373  
 Return program file name being executed  
     HPMYPROGRAM, 4-389  
 Return program file name that called intrinsic

- HPMYFILE, 4-387
- Return program to nonprivileged mode
  - GETUSERMODE, 4-286
- Returns information about a file
  - FFILEINFO, 4-131
- Returns message corresponding to FCHECK
  - error number
  - FERRMSG, 4-130
- Return source of activation call for process
  - GETORIGIN, 4-279
- Return status for parent/child process
  - GETPROCINFO, 4-284
- Return system defined JCW value
  - GETJCW, 4-277
- Return system time
  - CLOCK, 4-31
- Return system timer information
  - TIMER, 4-608
- Return user-supplied information
  - GETINFO, 4-275
- Return volume information
  - HPVOLINFO, 4-425
- RIN, determine process PIN that locked local
  - LOCKRINOWNER, 4-466
- RIN, lock global
  - LOCKGLORIN, 4-462
- RIN, lock local
  - LOCKLOCRIN, 4-464
- RINs, acquire local
  - GETLOCRIN, 4-278
- RINs, free from allocation to job
  - FREELOCRIN, 4-240
- RIN, unlock global
  - UNLOCKGLORIN, 4-610
- RIN, unlock local
  - UNLOCKLOCRIN, 4-611
- Routine header, 1-3

## **S**

- Scan and move character strings
  - NLSCANMOVE, 4-539
- Scratch file, close and restore data stack
  - SORTEND, 4-585
- SEARCH
  - Intrinsic, 4-578
  - Search array for specified entry/name, 4-578
- Search array for specified entry/name
  - SEARCH, 4-578
- Search JCW table and return value
  - FINDJCW, 4-166
- Search String
  - NLFINDSTR, 4-500
- SENDMAIL
  - Intrinsic, 4-580
  - Send mail to another process, 4-580

- Send mail to another process
  - SENDMAIL, 4-580
- Session information, access
  - JOBINFO, 4-452
- Session, initiate on specified terminal
  - STARTSESS, 4-599
- Session, place in break mode
  - CAUSEBREAK, 4-28
- Set bits in stream JCW
  - SETJCW, 4-584
- Set condition code for calling process
  - HPSETCCODE, 4-399
- SETDUMP
  - Arm system debugger from process abort, 4-582
  - Intrinsic, 4-582
- SETJCW
  - Intrinsic, 4-584
  - Set bits in stream JCW, 4-584
- Set logical record pointer for disk file
  - FPOINT, 4-221
- Setting, define parameter bit, 2-2
- Set value of session-level variable
  - HPCIPUTVAR, 4-305
- shareable device, create file
  - HPFOPEN, 4-331
- share data pointer
  - HPFDUPLICATE, 4-327
- Size, alter data segment
  - ALTDSEG, 4-10
- SL procedure, load compatibility mode
  - LOADPROC, 4-460
- SL procedure, unload compatibility mode
  - UNLOADPROC, 4-609
- Software events, 2-12
- Software interrupts, enable/disable
  - FINTSTATE, 4-169
- Software library trap handling procedure,
  - arm/disarm
  - XLIBTRAP, 4-624
- Software traps, 2-12
- SORTEND
  - Close scratch file and restore data stack, 4-585
  - Intrinsic, 4-585
- SORTERRORMESS
  - Intrinsic, 4-586
  - Retrieve and print message if fatal error, 4-586
- SORTINIT
  - Initiate SORT program, 4-587
  - Intrinsic, 4-587
- SORTINPUT
  - Intrinsic, 4-592
  - Specify how records are supplied to SORT program, 4-592

SORT operation end, release work area  
     HPSORTEND, 4-403  
 SORTOUTPUT  
     Intrinsic, 4-593  
     Specify how records are output from SORT  
     program, 4-593  
 SORT, pass records to  
     HPSORTINPUT, 4-411  
 SORT program, how records are output  
     SORTOUTPUT, 4-593  
 SORT program, how records are supplied  
     SORTINPUT, 4-592  
 SORT program, initiate  
     SORTINIT, 4-587  
 SORT program, print statistics on \$STDLIST  
     SORTSTAT, 4-595  
 SORT, retrieve record from  
     HPSORTOUTPUT, 4-412  
 SORTSTAT  
     Intrinsic, 4-595  
     Print SORT program statistics on \$STDLIST,  
     4-595  
 SORT subsystem, initialize  
     HPSORTINIT, 4-406  
 SORTTITLE  
     Intrinsic, 4-596  
     Print version number/title on \$STDLIST,  
     4-596  
 Source code extensibility, 1-3  
 Source of activation call for process, return  
     GETORIGIN, 4-279  
 Specify how records are output from SORT  
     program  
     SORTOUTPUT, 4-593  
 Specify how records are supplied to SORT  
     program  
     SORTINPUT, 4-592  
 Specify record output from MERGE program  
     MERGEOUTPUT, 4-481  
 Spooled device file, control operations  
     FDEVICECONTROL, 4-108  
 STACKDUMP  
     Execute full stack trace, 4-597  
     Intrinsic, 4-597  
 Stack, read logical record from file  
     FREAD, 4-223  
 Stack trace, execute full  
     STACKDUMP, 4-597  
 STARTSESS  
     Initiate session on specified terminal, 4-599  
     Intrinsic, 4-599  
 Statistics for SORT, print on \$STDLIST  
     HPSORTSTAT, 4-414  
 Statistics, print MERGE on \$STDLIST  
     HPMERGESTAT, 4-384  
 Status for parent/child process, return  
     GETPROCINFO, 4-284  
 Status information for KSAM file  
     FGETKEYINFO, 4-164  
 Status information returned, file  
     FGETINFO, 4-155  
 Status parameter error, 2-9  
 String conversion, ASCII to EBCDIC  
     CTRANSLATE, 4-48  
 String conversion, EBCDIC to ASCII  
     CTRANSLATE, 4-48  
 String conversion, EBCDIK to JIS  
     CTRANSLATE, 4-48  
 String conversion, JIS to EBCDIK  
     CTRANSLATE, 4-48  
 String, convert ASCII to 16-bit binary  
     BINARY, 4-19  
 Strings, compare of different length  
     NLKEYCOMPARE, 4-528  
 String, translate from/to EBCDIC  
     NLTRANSLATE, 4-548  
 Subsystem break trap handling procedure,  
     arm/disarm  
     XCONTRAP, 4-622  
 SUSPEND  
     Intrinsic, 4-604  
     Suspend a process, 4-604  
 Suspend a process  
     SUSPEND, 4-604  
 Suspend calling process for time specified  
     PAUSE, 4-552  
 SWITCHDB  
     Change DB register pointer, 4-606  
     Intrinsic, 4-606  
 Switch register pointer  
     SWITCHDB, 4-606  
 Switch to CM, get plabel  
     HPLOADCMPROCEDURE, 4-371  
 Syntax  
     Format, 2-3  
     Intrinsic call description, 2-3  
 System access, services, 1-1  
 System catalog error message, get or display  
     HPERRMSG, 4-321  
 System debugger, arm call from process abort  
     HPSETDUMP, 4-401  
 System debugger, arm from process abort  
     SETDUMP, 4-582  
 System debugger, disarm from process abort  
     HPRESETDUMP, 4-394  
 System-defined files, assign and create process  
     CREATEPROCESS, 4-43  
 System defined JCW value, return  
     GETJCW, 4-277  
 System intrinsics

- Declaration, 2-1
- System resources, accessing, 1-1
- System supervisor (OP)
  - Capability, 2-14
- System-supplied procedures, 1-1
- System time, return
  - CLOCK, 4-31
- System timer information, return
  - TIMER, 4-608
- System trap handling procedure, arm/disarm
  - XSYSTRAP, 4-626

## T

- Table search, return JCW value
  - FINDJCW, 4-166
- Tape file, move record pointer
  - FSPACE, 4-250
- Tape, read logical record
  - FREADBACKWARD, 4-226
- Terminal, initiate specified session
  - STARTSESS, 4-599
- TERMINATE
  - Delete calling process, 4-607
  - Intrinsic, 4-607
- Terminate access to a file
  - FCLOSE, 4-79
- Test mailbox status
  - MAIL, 4-472
- Time and date, return
  - DATELINE, 4-52
- Time, convert from character string to numeric
  - NLCONVCLOCK, 4-493
- Time, format numeric as character string
  - NLFMTCLOCK, 4-504
- Time, format through language dependent
  - template
  - NLFMTDATE, 4-508
- Time of day, format
  - FMTCLOCK, 4-186
  - FMTDATE, 4-187
- TIMER
  - Intrinsic, 4-608
  - Return system timer information, 4-608
- Time, return actual system
  - CLOCK, 4-31
- Title information, print on \$STDLIST
  - HPMERGETITLE, 4-386
- Title, print on \$STDLIST
  - HPSORTTITLE, 4-416
  - MERGETITLE, 4-483
  - SORTTITLE, 4-596
- Transaction, mark beginning of user logging
  - BEGINLOG, 4-17
- Translate string from/to EBCDIC
  - NLTRANSLATE, 4-548

- Trap, enable/disable
  - HPENBLTRAP, 4-316
- Trap handling procedure, arm/disarm
  - XARITRAP, 4-617
  - XCONTRAP, 4-622
  - XLIBTRAP, 4-624
- Trap, intrinsic error, 2-12

## U

- UNLOADPROC
  - Dynamically unload compatibility mode SL
    - procedure, 4-609
    - Intrinsic, 4-609
  - Unload target procedure
    - HPUNLOADCMPROCEDURE, 4-423
- Unlock file, dynamically
  - FUNLOCK, 4-252
- Unlock global RIN
  - UNLOCKGLORIN, 4-610
- UNLOCKGLORIN
  - Intrinsic, 4-610
  - Unlock global RIN, 4-610
- Unlock local RIN
  - UNLOCKLOCRIN, 4-611
- UNLOCKLOCRIN
  - Intrinsic, 4-611
  - Unlock local RIN, 4-611
- Update logical record in disk file
  - FUPDATE, 4-253
- User defined command image, parse
  - MYCOMMAND, 4-484
- User file label, read
  - FREADLABEL, 4-234
- User file label, write
  - FWRITELABEL, 4-265
- User Logging facility, access
  - OPENLOG, 4-550
- user logging facility, close access
  - CLOSELOG, 4-32
- User Logging file, provide information
  - LOGINFO, 4-468
  - LOGSTATUS, 4-470
- User logging (LG)
  - Capability, 2-14
- User Logging memory buffer, flush contents
  - FLUSHLOG, 4-184
- user logging transaction, mark beginning
  - BEGINLOG, 4-17
- user logging transaction, mark end
  - ENDLOG, 4-64
- User Logging, write record
  - WRITELOG, 4-615
- User, return information about
  - WHO, 4-612
- User-supplied information, return

- GETINFO, 4-275
- Using
  - Itemnum and item pairs, 2-4
- Using intrinsics, 2-1
- USL file, change length
  - EXPANDUSLF, 4-66
- USL file, delete inactive entries
  - CLEANUSL, 4-29
- USL file, initialize to empty state
  - INITUSLF, 4-446
- Utility functions, request, 1-1

**V**

- Validate and parse file designator
  - FPARSE, 4-218
- Value of session-level variable, set
  - HPCIPUTVAR, 4-305
- Value, retrieve session-level
  - HPCIGETVAR, 4-301
- Value returned, search JCW table
  - +INDJCW, 4-166
- Value, return system defined JCW
  - GETJCW, 4-277
- Variable, set value
  - HPCIPUTVAR, 4-305
- Variable table, delete entry
  - HPCIDELETEVAR, 4-300
- Variable value, retrieve
  - HPCIGETVAR, 4-301
- Version number, print on \$STDLIST
  - HPMERGETITLE, 4-386
  - MERGETITLE, 4-483
  - SORTTITLE, 4-596
- Version, print on \$STDLIST
  - HPSORTTITLE, 4-416

**W**

- WHO
  - Intrinsic, 4-612
  - Return information about user, 4-612

- Work area, release and end merge
  - HPMERGEEND, 4-375
- Write file
  - ready, 4-395
- WRITELOG
  - Intrinsic, 4-615
  - Write record to User Logging file, 4-615
- Write logical/physical record from stack to file
  - FWRITE, 4-255
- Write logical record from stack to disk file
  - FWRITEDIR, 4-263
- Write record to User Logging file
  - WRITELOG, 4-615
- Write user file label
  - FWRITELABEL, 4-265

**X**

- XARITRAP
  - Arm/disarm arithmetic trap handling
    - procedure, 4-617
  - Intrinsic, 4-617
- XCONTRAP
  - Arm/disarm subsystem break trap handling
    - procedure, 4-622
  - Intrinsic, 4-622
- XLIBTRAP
  - Arm/disarm software library trap handling
    - procedure, 4-624
  - Intrinsic, 4-624
- XSYSTRAP
  - Arm/disarm system trap handling procedure,
    - 4-626
  - Intrinsic, 4-626

**Z**

- ZSIZE
  - Alter DB to Z area of compatibility mode
    - stack, 4-628
  - Intrinsic, 4-628