

Addendum to AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q

086-000195-00

This addendum updates your manual 093-000542-02. Please see "Updating Your Manual." If you are running AOS/VS Revision 7.69, do not insert this addendum, which becomes effective with AOS/VS Revision 7.70.

Copyright ©Data General Corporation, 1992

All Rights Reserved

Unpublished – all rights reserved under the copyright laws of the United States.

Printed in the United States of America

Revision 00, June 1992

Licensed Material – Property of Data General Corporation

Ordering No. 086-000195

Notice

DATA GENERAL CORPORATION (DGC) HAS PREPARED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF DGC; AND THE CONTENTS OF THIS MANUAL SHALL NOT BE REPRODUCED IN WHOLE OR IN PART NOR USED OTHER THAN AS ALLOWED IN THE DGC LICENSE AGREEMENT.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

This software is made available solely pursuant to the terms of a DGC license agreement, which governs its use.

Restricted Rights Legend: Use, duplication, or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at [DFARS] 252.227-7013 (October 1988).

Data General Corporation
4400 Computer Drive
Westboro, MA 01580

AViON, CEO, DASHER, DATAPREP, DESKTOP GENERATION, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, GENAP, INFOS, microNOVA, NOVA, PRESENT, PROXI, SWAT, TRENDVIEW, and WALKABOUT are U.S. registered trademarks of Data General Corporation; and AOSMAGIC, AOS/VSMAGIC, AROSE/PC, ArrayPlus, AV Object Office, AV Office, BaseLink, BusiGEN, BusiPEN, BusiTEXT, CEO Connection, CEO Connection/LAN, CEO Drawing Board, CEO DXA, CEO Light, CEO MAIL, CEO Object Office, CEO PXA, CEO Wordview, CEOwrite, COBOL/SMART, COMPUCALC, CSMAGIC, DASHER/One, DASHER/286, DASHER/286-12c, DASHER/286-12j, DASHER/386, DASHER/386-16c, DASHER/386-25, DASHER/386-25k, DASHER/386SX, DASHER/386SX-16, DASHER/386SX-20, DASHER/486-25, DASHER II/486-33TE, DASHER/LN, DATA GENERAL/One, DESKTOP/UX, DG/500, DG/AROSE, DGConnect, DG/DBUS, DG/Fontstyles, DG/GATE, DG/GEO, DG/HEO, DG/L, DG/LIBRARY, DG/UX, DG/XAP, ECLIPSE MV/1000, ECLIPSE MV/1400, ECLIPSE MV/2000, ECLIPSE MV/2500, ECLIPSE MV/3500, ECLIPSE MV/5000, ECLIPSE MV/5500, ECLIPSE MV/5600, ECLIPSE MV/7800, ECLIPSE MV/9300, ECLIPSE MV/9500, ECLIPSE MV/9600, ECLIPSE MV/10000, ECLIPSE MV/15000, ECLIPSE MV/18000, ECLIPSE MV/20000, ECLIPSE MV/30000, ECLIPSE MV/35000, ECLIPSE MV/40000, ECLIPSE MV/60000, FORMA-TEXT, GATEKEEPER, GDC/1000, GDC/2400, Intellibook, microECLIPSE, microMV, MV/UX, OpenMAC, PC Liaison, RASS, REV-UP, SLATE, SPARE MAIL, SUPPORT MANAGER, TEO, TEO/3D, TEO/Electronics, TURBO/4, UNITE, and XODIAC are trademarks of Data General Corporation.

Addendum to
AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q
086-000195-00

In the margins of replacement pages, a vertical bar indicates substantive technical change from 093-000542-02.

The addendum number appears on all pages in this addendum.

Updating Your Manual

This addendum (086-000195-00) to AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q introduces new information effective with AOS/VS II Release 2.20, and AOS/VS Release 7.70. It also includes minor corrections.

To update your copy of 093-000542-02, please remove manual pages and insert addendum pages as follows:

Remove

Title/Notice
iii through vi
old Contents
2-11 through 2-20
2-27/2-28
2-57/2-58
2-69/2-70
2-85/2-86
2-89/2-90
2-103/2-104
2-113/2-114
2-121/2-122
2-159 through 2-164
2-171 through 2-174
2-203 through 2-216
2-259/2-260
2-269 through 2-272
2-287/2-288
2-295 through 2-304
2-307/2-308
2-337/2-338
2-405 through 2-410
2-415 through 2-418
2-473/2-474
2-525 through 2-530
2-535 through 2-548
2-555/2-556

Insert

Title/Notice
iii through vi
new Contents
2-11 through 2-20
2-27/2-28
2-57/58 through 2-58.26
2-69/2-70
2-85/2-86
2-89/2-90
2-103/2-104
2-113/2-114
2-121/2-122
2-159 through 2-164
2-171 through 2-174
2-203 through 2-216
2-259/2-260
2-269 through 2-272
2-287/2-288
2-295 through 2-304.9
2-307/2-308
2-337/2-338
2-405 through 2-410
2-415 through 2-418
2-473/2-474
2-525 through 2-530
2-535 through 2-548
2-555/2-556

Remove

old Index

old Document Set

Insert

new Index

new Document Set

Where new material requires additional pages, the pages have a decimal and number suffix; for example 5-21.1, 5-22.2.

Insert this updating sheet immediately behind the new Title/Notice page.

AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q

093-000542-02

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000542

Copyright © Data General Corporation, 1988, 1990, 1991

All Rights Reserved

Unpublished – all rights reserved under the copyright laws of the United States.

Printed in the United States of America

Rev. 02, December, 1991

Licensed Material – Property of Data General Corporation

Notice

DATA GENERAL CORPORATION (DGC) HAS PREPARED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF DGC; AND THE CONTENTS OF THIS MANUAL SHALL NOT BE REPRODUCED IN WHOLE OR IN PART NOR USED OTHER THAN AS ALLOWED IN THE DGC LICENSE AGREEMENT.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

This software is made available solely pursuant to the terms of a DGC license agreement, which governs its use.

Restricted Rights Legend: Use, duplication, or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at [DFARS] 252.227-7013 (October 1988).

Data General Corporation
4400 Computer Drive
Westboro, MA 01580

AViiON, CEO, DASHER, DATAPREP, DESKTOP GENERATION, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, GENAP, INFOS, microNOVA, NOVA, PRESENT, PROXI, SWAT, TRENDVIEW, and WALKABOUT are U.S. registered trademarks of Data General Corporation; and AOSMAGIC, AOS/VSMAGIC, AROSE/PC, ArrayPlus, AV Object Office, AV Office, BaseLink, BusiGEN, BusiPEN, BusiTEXT, CEO Connection, CEO Connection/LAN, CEO Drawing Board, CEO DXA, CEO Light, CEO MAIL, CEO Object Office, CEO PXA, CEO Wordview, CEOWrite, COBOL/SMART, COMPUCALC, CSMAGIC, DASHER/One, DASHER/286, DASHER/286-12c, DASHER/286-12j, DASHER/386, DASHER/386-16c, DASHER/386-25, DASHER/386-25k, DASHER/386SX, DASHER/386SX-16, DASHER/386SX-20, DASHER/486-25, DASHER II/486-33TE, DASHER/LN, DATA GENERAL/One, DESKTOP/UX, DG/500, DG/AROSE, DGConnect, DG/DBUS, DG/Fontstyles, DG/GATE, DG/GEO, DG/HEO, DG/L, DG/LIBRARY, DG/UX, DG/XAP, ECLIPSE MV/1000, ECLIPSE MV/1400, ECLIPSE MV/2000, ECLIPSE MV/2500, ECLIPSE MV/3500, ECLIPSE MV/5000, ECLIPSE MV/5500, ECLIPSE MV/5600, ECLIPSE MV/7800, ECLIPSE MV/9300, ECLIPSE MV/9500, ECLIPSE MV/9600, ECLIPSE MV/10000, ECLIPSE MV/15000, ECLIPSE MV/18000, ECLIPSE MV/20000, ECLIPSE MV/30000, ECLIPSE MV/35000, ECLIPSE MV/40000, ECLIPSE MV/60000, FORMA-TEXT, GATEKEEPER, GDC/1000, GDC/2400, Intellibook, microECLIPSE, microMV, MV/UX, OpenMAC, PC Liaison, RASS, REV-UP, SLATE, SPARE MAIL, SUPPORT MANAGER, TEO, TEO/3D, TEO/Electronics, TURBO/4, UNITE, and XODIAC are trademarks of Data General Corporation.

AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q
093-000542-02

Revision History:
Original Release - October 1988
First Revision - February 1990
Second Revision - December 1991
Addendum 086-000195 - June 1992

Effective with:

AOS/VS, Rel. 7.70
AOS/VS II, Rel. 2.20
AOS/RT32, Rel. 5.01

A vertical bar in the outer margin of a page indicates substantive change from the previous revision of this manual.

Preface

The System Call Dictionary spans two manuals — one for system calls ?A through ?Q, and the other for system calls ?R through ?Z. Much information appears twice in these two manuals for your ease of use. For example, the table of contents and indexes are identical. Chapter 2 in both books has the same title, but their contents differ. Chapter 2 in manual 093–000543 is a continuation of Chapter 2 in manual 093–000542, and is paginated accordingly. Appendixes A and B follow Chapter 2 in the second manual.

This manual is intended for use by experienced assembly language programmers. Experienced high–level language programmers can also use this manual to create programs that make direct calls to the operating systems.

Organization

This manual is organized as follows:

- Chapter 2** begins with a summary table of all AOS/VS and AOS/RT32 system calls, followed by detailed descriptions of all the system calls whose names begin with ?A through ?Q.
- Appendix A** contains 12 program sets that illustrate AOS/VS and AOS/RT32 system calls. We have written 11 of the program sets in assembly language and the twelfth in FORTRAN 77. The Appendix is located at the end of the complementary manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z* (093–000543).
- Appendix B** describes the format of the *system log (SYSLOG) file*, into which both AOS/VS and AOS/VS II and privileged processes can write records that log the occurrence of certain events. The Appendix is located at the end of the complementary manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z* (093–000543).

Related Documentation

As mentioned earlier, the complement of this manual is *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z* (093–000543). The following documents are ancillary to both manuals.

- *AOS/VS System Concepts* (093–000335)
- *Introduction to AOS/RT32* (069–400061)
- *AOS/VS and AOS/VS II Error and Status Messages* (093–000540)
- *AOS/VS and AOS/VS II Glossary* (069–000231)

AOS/VS System Concepts and Introduction to AOS/RT32, listed at the beginning of this section, contain a general description of operating system calls and how to use them. This manual and its companion system call dictionary manual contain detailed descriptions of each AOS/VS and AOS/RT32 call. For your convenience, the system call descriptions in the two dictionaries are in alphabetical order.

The Documentation Set, after the index in each manual, contains a complete annotated list of AOS/VS and AOS/VS II manuals.

If you are not experienced with assembly language, we suggest that you read the following manuals before you read this book:

- *Fundamentals of Small Computer Programming* (093-000090), which provides a general introduction to Data General computers.
- *AOS/VS Macroassembler (MASM) Reference Manual* (093-000242), which gives detailed information about the syntax of AOS/VS assembly language and about the Macroassembler utility.
- *ECLIPSE® MV/Family (32-Bit) Systems Principles of Operation* (014-001371), which explains the processor-independent concepts and functions of ECLIPSE® MV/Family systems to assembly language programmers.
- *ECLIPSE® MV/Family (32-Bit) Systems Instruction Dictionary* (014-001372), which explains each instruction in the ECLIPSE MV/Family instruction set to assembly language programmers. Processor-dependent information, available in machine-specific supplements, complements this and the previous manual. An example of such information is found in the manual *ECLIPSE® MV/20000™ Series Systems Principles of Operation Supplement* (014-001169).
- *ECLIPSE® MV/Family Instruction Reference Booklet* (014-000702), which provides a brief summary of the instruction set and register information. The reference booklet lists each instruction by assembler-recognizable mnemonic with a shorthand description of its function.
- *FORTTRAN 77 Environment Manual (AOS/VS)* (093-000288).

Update and Release Notices

Certain features of the operating systems may change from revision to revision. Therefore, please refer to the current Release Notice for the most up-to-date information about functional changes and enhancements. The Release Notice is usually in the utilities directory (:UTIL) on your system. The filename of the AOS/VS Model 3900 Update Notice is 078_000105_**, that of the AOS/VS II Release Notice is 085_000930_**. Suffixes (**) change with each revision. Your system manager should be able to tell you the exact pathname of the Release Notice.

The AOS/VS and AOS/VS II Release and Update Notices contain the latest details about all the system software, including enhancements and changes, notes and warnings. Notices are supplied both as printed listings and as disk files that you can print. The manuals and the Notices comprise the documentation for the system calls for AOS/VS Revision 7.69, and for AOS/VS II Revision 2.10. There are no documentation-changes files for this manual.

You should read the Update and Release Notices. If you want to know the features of a release, or have problems with a release, read the notice for solutions. The notices assume that you know the operating system well — so parts of the notices may be difficult to understand until you *do* know the system.

The Newsletter

Finally, you will find the *AOS/VS Monthly Newsletter* a useful source of information on the latest enhancements to both AOS/VS and AOS/VS II.

Reader Please Note

Throughout this manual we use the following format conventions:

COMMAND required *[optional]* ...

Where	Means
COMMAND	You must enter the command (or its accepted abbreviation) as shown.
required	You must enter some argument (such as a filename). Sometimes, we use $\left\{ \begin{array}{l} \text{required}_1 \\ \text{required}_2 \end{array} \right\}$ which means you must enter one of the arguments. Do not type the braces; they only set off the choices.
<i>[optional]</i>	You have the option of entering this argument. Do not type the brackets; they only identify the argument as an option.
...	You may repeat the preceding entry.

Standard Symbols

Additionally, we use the following symbols:

Symbol	Means
↵	Press the New Line, Carriage Return (CR), or Enter key on your terminal keyboard.
)	The CLI prompt.
< >	Angle brackets indicate the paraphrase of an argument or statement. (You supply the actual argument or statement.)
*	One asterisk indicates multiplication. For example, 2*3 means 2 multiplied by 3.
**	Two asterisks indicate exponentiation. For example, 2**3 means 2 raised to the third power.
OS	The operating system in the accumulator I/O, figure, and table categories.

Unless the text supplies a specific radix (as it often does), all memory addresses are octal values and all other numbers are decimal values. To explicitly specify a decimal number, we sometimes use a period after the last digit. To explicitly specify an octal number, we sometimes use the phrases *octal value* or *base eight*. For example, the phrase “a baker’s dozen cookies” has traditionally meant 13. = 15 base eight cookies.

In this manual, AOS/VS means AOS/VS, AOS/VS II, or both, unless otherwise noted.

Finally, in examples we use

This typeface to show your entry.

This typeface to show system queries and responses.

This typeface to show listings and status displays.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Customer Support Center (CSC) by calling 1-800-DG-HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive *FOCUS* monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1-800-253-3902 or 1-508-443-3330.

End of Preface

Contents

Chapter 1 — Introducing the System Calls

Standard Format for System Calls	1-2
Parameter Packets	1-3
Parametric Coding	1-3
Other System Call Input	1-4
Reserved Symbols	1-5
A Complete Example — Assembly Language	1-5
General Step	1-5
Program Listing	1-6
Program Construction	1-8
Program Execution	1-8
Error Codes	1-9
High-Level Language Interface	1-10

Chapter 2 — AOS/VS, AOS/VS II, and AOS/RT32 System Calls

Summary Table	2-2
---------------------	-----

?A Through ?Q

?ALLOCATE	Allocates disk blocks.	2-15
?ASSIGN	Assigns a character device to a process.	2-17
?AWIRE	Changes the wiring characteristics of the Agent.	2-18
?BLKIO	Performs (reads/writes) block I/O.	2-19
?BLKPR	Blocks a process.	2-26
?BNAME	Determines whether process name/queue name is on local or remote host.	2-28
?BRKFL	Terminates a process and creates a break file	2-29
?CDAY	Converts a scalar date value.	2-31
?CGNAM	Gets a complete pathname from a channel number.	2-32
?CHAIN	Passes control from a Ring 7 caller to a new program.	2-33
?CKVOL	Checks volume identifier of a labeled magnetic tape.	2-35
?CLASS	Gets or sets class IDs.	2-36
?CLOSE	Closes an open channel.	2-38
?CLRDV	Clears a device.	2-40
?CLSCHED	Enables, disables, or examines class scheduling.	2-42
?CLSTAT	Returns class scheduling statistics.	2-45
?CMATRIX	Gets or sets the class matrix.	2-51
?CON	Becomes a customer of a specified server.	2-56
?CONFIG	Display or reset current MRC routes	2-58
?CONINFO	Request for addressing information on a terminal or console	2-58.6
?CPMAX	Sets maximum size for a control point directory (CPD).	2-58.26
?CREATE	Creates a file or directory.	2-60
?CRUDA	Creates a user data area (UDA).	2-70

?A Through ?Q

?CTERM	Terminates a customer process.	2-72
?CTOD	Converts a scalar time value.	2-74
?CTYPE	Changes a process type.	2-75
?DACL	Sets, clears, or examines a default access control list.	2-77
?DADID	Gets the PID of a process's father.	2-79
?DCON	Breaks a connection (disconnects) in Ring 7.	2-80
?DDIS	Disables access to all devices.	2-81
?DEASSIGN	Cancels a character device.	2-82
?DEBL	Enables access to all devices.	2-83
?DEBUG	Calls the Debugger utility.	2-84
?DELAY	Suspends a 16-bit task for a specified interval (16-bit processes only).	2-85
?DELETE	Deletes a file entry.	2-86
?DFRSCH	Disables task rescheduling and indicates prior state of rescheduling.	2-88
?DIR	Changes the working directory.	2-89
?DQTSK	Removes from the queue one or more previously queued tasks.	2-90
?DRCON	Breaks a connection (disconnects).	2-92
?DRSCH	Disables scheduling.	2-93
?ENBRK	Enables a break file.	2-94
?ENQUE	Sends a message to IPC and spooler files.	2-98
?ERMSG	Reads the error message file.	2-99
?ERSCH	Enables multitask scheduling for the calling process.	2-102
?ESFF	Flushes shared file memory pages to disk.	2-103
?EXEC	Requests a service from EXEC.	2-104
?EXPO	Sets, clears, or examines execute-protection status.	2-141
?FDAY	Converts date to a scalar value.	2-143
?FEDFUNC	Interfaces to File Editor (FED) utility.	2-144
?FEOV	Forces end-of-volume on labeled magnetic tape.	2-148
?FIDEF	Defines a fast user device.	2-149
?FIXMT	Transmits a message from an interrupt service routine in Ring 0.	2-157
?FLOCK	Locks an object.	2-159
?FLUSH	Flushes the contents of a shared page to disk.	2-162
?FSTAT	Gets file status information.	2-163
?FTOD	Converts time of day to a scalar value.	2-171
?FUNLOCK	Unlocks an object.	2-172
?GACL	Gets a file entry's access control list (ACL).	2-174
?GBIAS	Gets the current bias factor values.	2-176
?GCHR	Reads device characteristics of a character device.	2-177
?GCLOSE	Closes a file previously opened for block I/O.	2-183
?GCPN	Gets the terminal port number.	2-185
?GCRB	Gets the base of the current resource (16-bit processes only).	2-186
?GDAY	Gets the current date.	2-187
?GDLM	Gets a delimiter table.	2-188
?GECHR	Get extended characteristics.	2-190
?GHRZ	Gets the frequency of the system clock.	2-201

?A Through ?Q

?GLINK	Gets the contents of a link entry.	2-202
?GLIST	Gets the contents of a search list.	2-203
?GMEM	Returns the number of undedicated memory pages.	2-204
?GNAME	Gets a complete pathname.	2-205
?GNFN	Lists a particular directory's entries.	2-207
?GOPEN	Opens a file for block I/O.	2-210
?GPID	Returns all active PIDs based on a host ID.	2-217
?GPORT	Returns the PID associated with a global port number.	2-219
?GPOS	Gets the current file-pointer position.	2-220
?GPRNM	Gets a program's pathname.	2-222
?GRAPHICS	Manipulates pixel maps.	2-223
?GRNAME	Returns complete pathname of generic file.	2-239
?GROUP	Changes a group access control list of a process.	2-240
?GSHPT	Lists the current shared partition size.	2-243
?GSID	Gets the system identifier.	2-244
?GTACP	Gets access control privileges.	2-245
?GTIME	Gets the time, date, and time zone.	2-247
?GTMES	Gets an initial IPC message.	2-250
?GTNAM	Returns symbol closest in value to specified input value.	2-256
?GTOD	Gets the time of day.	2-258
?GTRUNCATE	Truncates a disk file.	2-259
?GTSVL	Gets the value of a user symbol.	2-261
?GUHPI	Gets unique hardware processor identification.	2-263
?GUNM	Gets the username of a process.	2-265
?GVPID	Gets the virtual PID of a process.	2-266
?HNAME	Gets a hostname or host identifier.	2-267
?IDEF	Defines a user device.	2-269
?IDGOTO	Redirects a task's execution path.	2-278
?IDKIL	Kills a task specified by its TID.	2-279
?IDPRI	Changes the priority of a task specified by its TID.	2-280
?IDRDY	Readies a task specified by its TID.	2-281
?IDSTAT	Returns task status word (16-bit processes only).	2-282
?IDSUS	Suspends a task specified by its TID.	2-283
?IESS	Initializes an extended state save (ESS) area (16-bit processes only).	2-284
?IFPU	Initializes the floating-point unit.	2-285
?IHIST	Starts a histogram for a 16-bit process (16-bit processes only).	2-286
?ILKUP	Returns a global port number.	2-288
?IMERGE	Modifies a ring field within a global port number.	2-289
?IMSG	Receives an interrupt service message.	2-290
?INIT	Initializes a logical disk.	2-291
?INTWT	Defines a terminal interrupt task.	2-293
?IQTSK	Creates a queued task manager.	2-294
?IREC	Receives an IPC message.	2-295
?IRMV	Removes a user device.	2-304.9
?ISEND	Sends an IPC message.	2-305
?ISPLIT	Finds the owner of a port (including its ring number).	2-308

?A Through ?Q

?IS.R	Sends and then receives an IPC message.	2-309
?ITIME	Returns the OS-format internal time.	2-313
?IXIT	Exits from an interrupt service routine.	2-314
?IXMT	Transmits a message from an interrupt service routine.	2-315
?JPINIT	Initializes a job processor.	2-317
?JPMOV	Moves a job processor to a new logical processor.	2-320
?JPREL	Releases a job processor.	2-322
?JPSTAT	Gets the status of a job processor.	2-324
?KCALL	Keeps the calling resource and acquires a new resource (16-bit processes only).	2-328
?KHIST	Kills a histogram.	2-329
?KILAD	Defines a kill-processing routine.	2-330
?KILL	Kills the calling task.	2-331
?KINTR	Simulates keyboard interrupt sequences.	2-332
?KIOFF	Disables control-character terminal interrupts.	2-333
?KION	Re-enables control-character terminal interrupts.	2-334
?KWAIT	Waits for a terminal interrupt.	2-335
?LABEL	Creates a label for a magnetic tape or diskette.	2-336
?LDUINFO	Obtain logical disk information.	2-340
?LEFD	Disables LEF mode.	2-350
?LEFE	Enables LEF mode.	2-351
?LEFS	Returns the current LEF mode status.	2-352
?LMAP	Maps a lower ring.	2-353
?LOCALITY	Changes user locality.	2-354
?LOGCALLS	Logs system calls.	2-357
?LOGEV	Enters an event in the system log file.	2-360
?LPCLASS	Gets/sets logical processor class assignments.	2-362
?LPCREA	Creates a logical processor.	2-365
?LPDELE	Deletes a logical processor.	2-367
?LPSTAT	Gets the status of a logical processor.	2-369
?MAPDV	Maps a device into logical address space.	2-373
?MBFC	Moves bytes from a customer's buffer.	2-377
?MBTC	Moves bytes to a customer's buffer.	2-379
?MDUMP	Dumps the memory image from a user-specified ring to a file.	2-381
?MEM	Lists the current unshared memory parameters.	2-383
?MEMI	Changes the number of unshared pages in the logical address space.	2-384
?MIRROR	Mirrors and synchronizes LDU images.	2-386
?MPHIST	Starts a histogram on a uni- or multi-processor system (32-bit processes only).	2-394
?MYTID	Gets the priority and TID of the calling task.	2-399
?NTIME	Sets the time, date, and time zone.	2-400
?ODIS	Disables terminal interrupts.	2-403
?OEBL	Enables terminal interrupts.	2-404
?OPEN	Opens a file.	2-405
?OPER	Creates and maintains an operator interface.	2-424
?OPEX	Communicates between the current process and an operator process.	2-437

?A Through ?Q

?OVEX	Releases an overlay and returns (16-bit processes only).	2-509
?OVKIL	Exits from an overlay and kills the calling task (16-bit processes only).	2-510
?OVL0D	Loads and goes to an overlay (16-bit processes only).	2-511
?OVREL	Releases an overlay area (16-bit processes only).	2-513
?PCLASS	Gets a process's class and locality.	2-514
?PCNX	Passes a connection from one server to another in Ring 7.	2-516
?PIDS	Gets information about PIDs.	2-517
?PMPF	Permits access to a protected file.	2-519
?PNAME	Gets a full process name.	2-521
?PRCNX	Passes a connection from one server to another.	2-523
?PRDB/?PWRB	Performs physical block I/O.	2-525
?PRI	Changes the priority of the calling task.	2-530
?PRIPR	Changes the priority of a process.	2-531
?PRKIL	Kills all tasks of a specified priority.	2-533
?PROC	Creates a process.	2-534
?PROFILE	Performs a profile request.	2-551
?PRRDY	Readies all tasks of a specified priority.	2-558
?PRSUS	Suspends all tasks of a specified priority.	2-559
?PSTAT	Returns status information on a process.	2-560
?PTRDEVICE	Controls input from a pointer device.	2-567
?PWDCRYP	Performs a password data encryption request.	2-590
?PWRB	Performs physical block I/O.	2-592

Index

Document Set

Chapter 2 — AOS/VS, AOS/VS II, and AOS/RT32 System Calls (Continued)

?R Through ?Z

?RCALL	Releases one resource and acquires a new one (16-bit processes only).	2-594
?RCHAIN	Chains to a new procedure (16-bit processes only).	2-595
?RDB/?WRB	Performs (reads/writes) block I/O.	2-596
?RDUDA/?WRUDA	Reads/writes a user data area (UDA).	2-602
?READ/?WRITE	Performs (reads/writes) record I/O.	2-604
?REC	Receives an intertask message.	2-627
?RECNW	Receives an intertask message without waiting.	2-628
?RECREATE	Recreates a file.	2-629
?RELEASE	Releases an initialized logical disk.	2-630
?RENAME	Renames a file.	2-631
?RESCHED	Reschedules current time slice (32-bit processes only).	2-633

?R Through ?Z

?RESIGN	Resigns as a server.	2-634
?RETURN	Terminates the calling process and passes the termination message to the father.	2-635
?RINGLD	Loads a program file into a specified ring.	2-637
?RNAME	Determines whether a pathname contains a reference to a remote host.	2-639
?RNGPR	Returns the .PR filename for a ring.	2-640
?RNGST	Stops ?RINGLD from loading lower rings.	2-642
?RPAGE	Releases a shared page.	2-643
?RTODC	Reads the time-of-day conversion data.	2-645
?RUNTM	Gets runtime statistics on a process.	2-648
?SACL	Sets a new access control list (ACL).	2-650
?SATR	Sets or removes the permanent attribute for a file or directory.	2-653
?SBIAS	Sets the bias factors.	2-655
?SCHR	Sets a character device's characteristics.	2-656
?SCLOSE	Closes a file previously opened for shared access.	2-658
?SDAY	Sets the system calendar.	2-660
?SDBL	Disables a BSC line.	2-661
?SDLM	Sets a delimiter table.	2-662
?SDPOL	Defines a polling list or a poll-address/ select-address pair.	2-664
?SDRT/?SERT	Disables/re-enables a relative terminal.	2-667
?SEBL	Enables a BSC line.	2-669
?SECHR	Sets extended characteristics of a device.	2-679
?SEND	Sends a message to a terminal.	2-681
?SERMSG	Returns text for associated error code (16-bit processes only).	2-683
?SERT	Re-enables a relative terminal.	2-685
?SERVE	Becomes a server.	2-685
?SGES	Gets BSC error statistics.	2-686
?SIGNL	Signals another task.	2-688
?SIGWT	Signals another task and then waits for a signal.	2-690
?SINFO	Gets selected information about the current operating system.	2-692
?SLIST	Sets the search list for the calling process.	2-695
?SONS	Gets a list of son processes for a target PID.	2-696
?SOPEN	Opens a file for shared access.	2-699
?SOPPF	Opens a protected shared file.	2-701
?SPAGE	Performs a shared-page read.	2-704
?SPOS	Sets the position of the file pointer.	2-707
?SRCV	Receives data or a control sequence over a BSC line.	2-709
?SSHPT	Establishes a new shared partition.	2-718
?SSID	Sets the system identifier.	2-719
?SSND	Sends data or a control sequence over a BSC line.	2-720
?STMAP	Sets the data channel map.	2-729
?STOD	Sets the system clock.	2-732
?STOM	Sets the time-out value for a device.	2-733
?SUPROC	Enters, leaves, or examines Superprocess mode.	2-735

?R Through ?Z

?SUS	Suspends the calling task.	2-736
?SUSER	Enters, leaves, or examines Superuser mode.	2-737
?SYLOG	Manipulates the system log files.	2-739
?SYSPRV	Enters, leaves, or examines a privilege state.	2-744
?TASK	Initiates one or more tasks.	2-747
?TERM	Terminates a process.	2-754
?TIDSTAT	Returns status of target task (32-bit processes only).	2-756
?TLOCK	Protects a task from being redirected.	2-757
?TMSG	Defines the termination message format.	2-759
?TPID	Translates a PID.	2-769
?TPORT	Translates a local port number to its global equivalent.	2-770
?TRCON	Reads a task message from the process terminal.	2-771
?TRUNCATE	Truncates a file at the current position.	2-773
?TUNLOCK	Allows a task to be redirected.	2-774
?UBLPR	Unlocks a process.	2-776
?UIDSTAT	Returns the status of a task and an unambiguous identifier.	2-777
?UNWIND	Unwinds the stack and restores the previous environment (16-bit processes only).	2-778
?UNWIRE	Unwires pages previously wired.	2-779
?UPDATE	Flushes file descriptor information.	2-780
?VALAD	Validates a logical address.	2-781
?VALIDATE	Validates an area for Read or Write access.	2-782
?VCUST	Verifies a customer in Ring 7.	2-786
?VMEM	Changes the partition size of a process.	2-787
?VRCUST	Verifies a customer in a specified ring.	2-789
?VTFCREATE	Creates a Virtual Timer Facility timer.	2-790
?VTFKILL	Kills a Virtual Timer Facility timer.	2-797
?VTFMODIFY	Modifies a Virtual Timer Facility timer.	2-799
?VTFSUS	Suspends or Restarts a Virtual Timer Facility timer.	2-804
?VTFXIT	Exits from a Virtual Timer Facility interrupt routine.	2-806
?WALKBACK	Returns information about previous frames in the stack (16-bit processes only).	2-807
?WDELAY	Suspends a task for a specified time (32-bit processes only).	2-809
?WHIST	Starts a histogram (32-bit processes only).	2-810
?WINDOW	Manipulates windows.	2-813
?WIRE	Wires pages to the working set.	2-842
?WRB	Writes block I/O.	2-844
?WRITE	Writes record I/O.	2-844
?WRUDA	Writes a user data area (UDA).	2-844
?WTSIG	Waits for a signal from another task or process.	2-845
?WTVERR	Waits for a Virtual Timer Facility error message.	2-846
?WTVSIG	Waits for a Virtual Timer Facility signal.	2-848
?XCREATE	Creates a file or directory (extended).	2-849
?XFSTAT	Gets file status information (extended).	2-862
?XGTACP	Gets access control privileges (extended).	2-882
?XINIT	Initializes a logical disk (extended).	2-886
?XMT	Transmits an intertask message.	2-898

?R Through ?Z

?XMTW	Transmits an intertask message and waits for it to be received.	2-899
?XPSTAT	Returns extended status information on a process.	2-900

Appendix A — Sample Programs

Program Set 1 — HEAR.SR, SPEAK.SR, SON.SR	A-3
Program Set 2 — RUNTIME.SR	A-11
Program Set 3 — RINGLOAD.SR, INRING.SR, and GATE.ARRAY.SR	A-14
Program Set 4 — FILCREATE.SR	A-19
Program Set 5 — WRITE.SR	A-22
Program Set 6 — DLIST.SR	A-26
Program Set 7 — NEWTASK.SR	A-29
Program Set 8 — BOOMER.SR	A-32
Program Set 9 — TIMEOUT.SR	A-37
Program Set 10 — DIRCREATE.F77 and CHECK.F77	A-39
Program Set 11 — CREATE_WINDOW.SR	A-46
Program Set 12 — GRAPHICS_SAMPLE.SR	A-56

Appendix B — System Log Record Format

Reporting the Contents of the SYSLOG File	B-1
Reading the SYSLOG File	B-1
Record Header Format	B-2
SYSLOG Record Formats	B-3
Anatomy of a System Log File Record	B-20

Index

Document Set

Tables

Table

?A Through ?Q

2-1	Summary of AOS/VS and AOS/RT32 System Calls	2-3
2-2	Contents of ?BLKIO Packet	2-21
2-3	Contents of ?CLASS Packet	2-37
2-4	Contents of ?CLSCHED Packet	2-44
2-5	Contents of ?CLSTAT Main Packet	2-48
2-6	Contents of ?CLSTAT Subpacket	2-50
2-7	Contents of ?CMATRIX Main Packet	2-53
2-8	Contents of ?CMATRIX Subpacket	2-54
2-8.1.	Valid ?CONFIG_FUNCTION Function Codes	2-58.1
2-8.2	?CONFIG GET CURRENT ROUTE Function Subpacket Contents	2-58.3
2-8.3	?CONFIG RESET_MRD_CHANNEL Function Subpacket Contents	2-58.4
2-8.4	?CONFIG RESET_MRC_CONTROLLER Function Subpacket Contents	2-58.5
2-8.5	Contents of the ?CONINFO Packet	2-58.7
2-8.6	Input Values to ?CON_PKTUSER_FLGS Offset	2-58.8
2-8.7	?CON_RET_TYPES Return Buffer Console Types and Definitions	2-58.9
2-8.8	Contents of ?CON_TCP_RET_TYPE Packet	2-58.10
2-8.9	Contents of ?CON_XNS_RET_TYPE Packet	2-58.12
2-8.10	Contents of ?CON_CON_RET_TYPE Packet	2-58.13
2-8.11	Contents of ?CON_TNET_RET_TYPE Return Packet	2-58.14
2-8.12	Contents of ?CON_ITC_MIN_DATA Packet	2-58.15
2-8.13	Contents of ?CON_TSC_MIN_DATA Packet	2-58.16
2-8.14	Contents of ?CON_PVC_RET_TYPE Packet	2-58.17
2-8.15	?CON_PVC_RET_TYPE Subpacket Types and Definitions	2-58.18
2-8.16	Contents of ?CON_PVC_NAME Subpacket	2-58.19
2-8.17	Contents of ?CON_PVC_NAME_PORT Subpacket	2-58.20
2-8.18	Contents of ?CON_PVC_IP Subpacket	2-58.21
2-8.19	Contents of ?CON_PVC_IP_PORT Subpacket	2-58.22
2-8.20	Contents of ?CON_PVC_ETH Subpacket	2-58.23
2-8.21	Contents of ?CON_PVC_PORT Subpacket	2-58.24
2-8.22	Contents of ?CON_PVC_NET Subpacket	2-58.25
2-9	Valid ?CREATE File Types	2-61
2-10	Contents of ?CREATE IPC Packet	2-63
2-11	Contents of ?CREATE Directory Packet	2-65
2-12	Contents of ?CREATE Packet for Other File Types	2-67
2-13	Contents of ?ENBRK Packet	2-96
2-14	Flags for EXEC Functions ?XFXUN and ?XFXML	2-108
2-15	Contents of ?EXEC Packet for Tape Backup	2-111
2-16	Contents of ?EXEC Packet for Queue Requests	2-114
2-17	Contents of ?XFUSR Subpacket of ?EXEC System Call	2-125
2-18	Contents of AOS/VS ?EXEC Packet for Hold, Unhold, or Cancel Queue Requests	2-128
2-19	Contents of ?EXEC Packet for Status Information	2-129
2-20	Contents of ?EXEC Packet for Extended Status Information	2-130
2-21	Contents of Selected Offsets in the ?EXEC Packet for the ?XFMOD Function	2-135

Table**?A Through ?Q**

2-22	Queue Status Bit Definitions in Offset ?XQ1FG	2-136
2-23	?EXEC Queue Types in Offset ?XQQT	2-138
2-24	Contents of ?FEDFUNC Packet to Evaluate a FED String	2-147
2-25	Contents of Map Definition Entry	2-154
2-26	Contents of ?FLOCK Packet	2-161
2-27	Flags Returned in Offset ?SSTS	2-169
2-28	Contents of ?FUNLOCK Packet	2-173
2-29	Character Device Characteristics Words	2-178
2-30	Commonly Used Device Characteristics	2-181
2-33	Device Types for Rubout Echo and Cursor Controls	2-200
2-34	Contents of ?GNFN Packet	2-208
2-35	Filename Template Characters	2-208
2-36	Contents of ?GOPEN Packet for IPC Files	2-211
2-37	Contents of Standard ?GOPEN Packet	2-212
2-38	Option Flags for Offset ?ODF1	2-215
2-39	Valid Format Options (DPM disks)	2-216
2-40	Contents of ?GPID Packet	2-218
2-41	?GRAPHICS Function Codes	2-224
2-42	Contents of the ?GRAPHICS Main Packet	2-225
2-43	Contents of the ?GRAPH_OPEN_WINDOW_PIXELMAP Subpacket	2-227
2-44	Contents of the ?GRAPH_CREATE_MEMORY_PIXELMAP Subpacket	2-228
2-45	Contents of the ?GRAPH_PIXELMAP_STATUS Subpacket	2-230
2-46	Contents of the ?GRAPH_SET_CLIPRECTANGLE Subpacket	2-231
2-47	Contents of the ?GRAPH_MAP_PIXELMAP Subpacket	2-233
2-48	Contents of the ?GRAPHICS_SET_DRAW_ORIGIN Subpacket	2-237
2-49	Contents of the ?GRAPHICS_GET_DRAW_ORIGIN Subpacket	2-238
2-50	Contents of ?GROUP Packet	2-242
2-51	Contents of ?GTIME Packet	2-249
2-52	Contents of ?GTMES Packet	2-251
2-53	Input Parameters for Offset ?GREQ (Request Types)	2-252
2-54	Output from ?GTMES Requests	2-254
2-55	Contents of ?GTRUNCATE Packet	2-260
2-56	Contents of ?GUHPI Packet	2-264
2-57	Contents of Map Definition Entry	2-274
2-58	Structure of ?IHIST Array	2-287
2-59	Contents of ?INIT Packet	2-292
2-60	Contents of ?IREC Header	2-297
2-61	Termination Codes for 16-Bit Processes	2-298
2-62	Process Termination Codes in Offset ?IUFL for ?IREC and ?ISEND Headers	2-299
2-63	?TEXT Code Termination Messages Sent on an A-Type 32-Bit Process User Trap ..	2-301
2-63.1	?TRAP Termination Messages for A-Type 16-Bit Processes	2-303
2-63.2	Contents of Termination Message from a 32-bit B- or C-Type Process	2-304.2
2-64	Contents of Termination Message from a 16-bit B- or C-Type Process	2-304.6
2-65	Contents of ?ISEND Header	2-306
2-66	Contents of ?IS.R Header	2-311
2-67	Contents of ?JPINIT Packet	2-319
2-68	Contents of ?JPMOV Packet	2-321
2-69	Contents of ?JPREL Packet	2-323
2-70	Contents of ?JPSTAT Main Packet	2-325

Table**?A Through ?Q**

2-71	Contents of ?JPSTAT General Information Subpacket	2-326
2-72	Contents of ?JPSTAT Specific Information Subpacket	2-327
2-73	Contents of ?LABEL Packet	2-338
2-74	Contents of ?LDUINFO Main Packet	2-341
2-75	Contents of ?LDU_PKT Subpacket	2-344
2-76	?PIECE_PKT Subpacket Contents	2-348
2-77	Contents of ?LOCALITY Packet	2-356
2-78	Contents of ?LPCLASS Packet	2-364
2-79	Contents of ?LPCREA Packet	2-366
2-80	Contents of ?LPDELE Packet	2-368
2-81	Contents of ?LPSTAT Main Packet	2-370
2-82	Contents of ?LPSTAT General Information Subpacket	2-371
2-83	Contents of ?LPSTAT Specific Information Subpacket	2-372
2-84	Contents of ?MAPDV Packet	2-376
2-85	Contents of ?MIRROR Packet	2-388
2-86	Contents of 16-Bit ?MIRROR Packet	2-391
2-87	Contents of ?MPHIST Packet	2-397
2-88	Structure and Contents of ?MPHIST Histogram Array	2-398
2-89	Contents of ?NTIME Packet	2-402
2-90	Contents of ?OPEN Packet	2-408
2-91	File Creation Options for Offset ?ISTI	2-413
2-92	Common File Types You Can Create with ?OPEN	2-414
2-93	Contents of ?OPEN Extension Packet for Pipes	2-417
2-94	Contents of Labeled Magnetic Tape Packet Extension	2-420
2-95	Contents of ?OPER Main Packet	2-426
2-96	Contents of ?OPON Subpacket	2-427
2-97	Contents of ?OPOFF Subpacket	2-429
2-97	Contents of ?OPOFF Subpacket	2-430
2-98	Contents of ?OPSEND Subpacket	2-431
2-99	Contents of ?OPRCV Subpacket	2-433
2-100	Contents of ?OPRESP Subpacket	2-434
2-101	Contents of ?OPINFO Subpacket	2-436
2-102	Contents of ?OPEX Main Packet	2-439
2-103	Contents of Access Command Subpacket	2-442
2-104	Contents of Align Command Subpacket	2-443
2-105	Contents of Batch_List Command Subpacket	2-444
2-106	Contents of Batch_Output Command Subpacket	2-445
2-107	Contents of Binary Command Subpacket	2-446
2-108	Contents of Brief Command Subpacket	2-447
2-109	Contents of Cancel Command Subpacket	2-448
2-110	Contents of Consolestatus Command Subpacket	2-450
2-111	Contents of Continue Command Subpacket	2-451
2-112	Contents of CPL Command Subpacket	2-452
2-113	Contents of Create Command Subpacket	2-453
2-114	?OPEX Queue Types in Offset ?ZCRQ	2-453
2-115	Contents of Defaultforms Command Subpacket	2-454
2-116	Contents of Disable Command Subpacket	2-455
2-117	Contents of Dismounted Command Subpacket	2-456

Table

?A Through ?Q

2-118	Contents of Elongate Command Subpacket	2-457
2-119	Contents of Enable Command Subpacket	2-459
2-120	Contents of Even Command Subpacket	2-460
2-121	Contents of Flush Command Subpacket	2-461
2-122	Contents of Forms Command Subpacket	2-462
2-123	Contents of Halt Command Subpacket	2-463
2-124	Contents of Headers Command Subpacket	2-464
2-125	Contents of Hold Command Subpacket	2-465
2-126	Contents of Limit Command Subpacket	2-466
2-127	Contents of Logging Command Subpacket	2-468
2-128	Contents of LPP Command Subpacket	2-469
2-129	Contents of Mapper Command Subpacket	2-470
2-130	Contents of Mounted Command Subpacket	2-472
2-131	Contents of Mountstatus Command Subpacket	2-474
2-132	Contents of Operator Command Subpacket	2-476
2-133	Contents of Pause Command Subpacket	2-477
2-134	Contents of Premount Command Subpacket	2-478
2-135	Contents of Priority Command Subpacket	2-479
2-136	Contents of Prompts Command Subpacket	2-480
2-137	Contents of Qpriority Command Subpacket	2-482
2-138	Contents of Refused Command Subpacket	2-483
2-139	Contents of Release Command Subpacket	2-484
2-140	Contents of Restart Command Subpacket	2-485
2-141	Contents of Silence Command Subpacket	2-486
2-142	Contents of Spoolstatus Command Subpacket	2-488
2-143	Contents of Stack Command Subpacket	2-491
2-144	Contents of Start Command Subpacket	2-493
2-145	Contents of Status Command Subpacket	2-496
2-146	Contents of Stop Command Subpacket	2-499
2-147	Contents of Trailers Command Subpacket	2-500
2-148	Contents of Unhold Command Subpacket	2-501
2-149	Contents of Unitstatus Command Subpacket	2-503
2-150	Contents of Unlimit Command Subpacket	2-504
2-151	Contents of Unsilence Command Subpacket	2-505
2-152	Contents of Subpacket User-Command	2-507
2-153	Contents of Verbose Command Subpacket	2-508
2-154	Contents of ?PCLASS Packet	2-515
2-155	Contents of ?PIDS Packet	2-518
2-156	Contents of ?PMTPF Packet	2-520
2-157	Contents of ?PRDB/?PWRB Packet	2-527
2-158	?PRDB/?PWRB Packet: Controller Status Words	2-528
2-159	Error Reports Returned in ?PRDB/?PWRB Offsets	2-529
2-160	Contents of ?PROC Packet	2-538
2-161	Privilege Bits in Offset ?PPRV	2-542
2-162	Contents of ?PROC Parameter Packet Extension for AOS/VS and AOS/RT32	2-547
2-163	Contents of ?PROC Parameter Packet Extension for AOS/VS II	2-548
2-164	Contents of ?PROFILE Parameter Packet	2-553
2-165	Contents of ?PSTAT Parameter Packet	2-563
2-166	?PTRDEVICE Function Codes	2-568

Table ?A Through ?Q

2-167	Contents of the ?PTRDEVICE Main Packet	2-570
2-168	Flags for Selecting Pointer Events (Flag Word ?PTRDEV_SET_EVTS.EVTS)	2-575
2-169	Flags for Selecting Buttons (Flag Word ?PTRDEV_SET_EVTS.BTNS)	2-576
2-170	Contents of the ?PTRDEV_SET_DELTA Subpacket	2-577
2-171	Contents of the ?PTRDEV_LAST_EVENT Subpacket	2-578
2-172	Contents of the ?PTRDEV_SET_POINTER Subpacket	2-580
2-173	Contents of the ?PTRDEV_GET_PTR_STATUS Subpacket	2-582
2-174	Flags for Each Possible Pointer Device Event (Flag Word ?PTRDEV_GSTATUS.EVTS)	2-584
2-175	Contents of the ?PTRDEV_GENERATE_EVENT Subpacket	2-586
2-176	Contents of the ?PTRDEV_GET_PTR_LOCATION Subpacket	2-588
2-177	Contents of the ?PTRDEV_GET_TABLET_LOCATION Subpacket	2-589
2-178	Contents of ?PWDCRYP Packet	2-591

?R Through ?Z

2-179	Contents of ?RDB/?WRB Packet	2-598
2-180	Contents of the Standard ?READ/?WRITE Packet	2-608
2-181	Contents of Screen-Management Packet Extension	2-613
2-182	Contents of Selected Field Translation Packet Extension	2-618
2-183	Contents of the New Screen Management Packet	2-623
2-185	Supported (Y) and Unsupported (N) New Screen Management Packet Features in AOS/VS	2-626
2-186	Contents of ?RNGPR Packet	2-641
2-187	Contents of ?RTODC Packet	2-647
2-188	Contents of ?SEBL Packet	2-671
2-189	BSC Protocol Data-Link Control Characters (DLCC)	2-674
2-190	Contents of the ?SERMSG Packet	2-684
2-191	Contents of ?SGES Packet	2-687
2-192	Contents of ?SONS Packet	2-698
2-193	Contents of ?SOPPF Packet	2-702
2-194	Contents of ?SPAGE Packet	2-705
2-195	File-Pointer Settings	2-708
2-196	Contents of ?SRCV Packet	2-711
2-197	Masks Returned on ?SRCV System Calls	2-716
2-198	Contents of ?SSND Packet	2-722
2-199	?SSND Call Types	2-726
2-200	Contents of ?SYSRV Packet	2-746
2-201	Contents of Standard Task Definition Packet	2-749
2-202	Contents of Extended Task Definition Packet	2-752
2-203	Contents of Termination Message a 32-bit Process Receives	2-762
2-204	Contents of Termination Message a 16-bit Process Receives	2-766
2-205	?VALIDATE Functions and Their Codes	2-783
2-206	Contents of ?VMEM Packet	2-788
2-207	Contents of ?VTFCREATE Packet	2-792
2-208	Contents of ?VTFKILL Packet	2-798
2-209	Contents of ?VTFMODIFY Packet	2-801
2-210	Histogram Array Structure	2-812
2-211	?WINDOW Function Codes	2-814
2-212	Contents of the ?WINDOW Main Packet	2-817

Table**?R Through ?Z**

2-213	Contents of the ?WIN_CREATE_WINDOW Subpacket	2-819
2-214	Contents of the ?WIN_DEFINE_PORTS Subpacket	2-824
2-215	Flags in Flag Word ?WIN_SINT.FLAGS	2-829
2-216	Border Types (Offset ?WIN_SINT.BORDER_TYPE)	2-829
2-217	Flags in the ?WIN_GET_USER_INTERFACE Flag Word (Flag Word ?WIN_GINT.FLAGS)	2-830
2-218	Border Types (Offset ?WIN_GINT.BORDER_TYPE)	2-831
2-219	Contents of the ?WIN_GTITLE Subpacket	2-832
2-220	Contents of the ?WIN_WINDOW_STATUS Subpacket	2-835
2-221	Contents of the ?WIN_DEVICE_STATUS Subpacket	2-839
2-222	Contents of ?XCREATE Main Packet	2-850
2-223	Valid ?XCREATE File Types	2-852
2-224	Contents of ?XCREATE Time-Block Subpacket	2-853
2-225	Contents of ?XCREATE Other Subpacket	2-854
2-226	Contents of ?XCREATE Directory Subpacket	2-856
2-227	Contents of ?XCREATE IPC Subpacket	2-857
2-228	Contents of ?XCREATE Link Subpacket	2-857
2-229	Valid ?XFSTAT File Types	2-863
2-230	?XFSTAT IPC File Status	2-867
2-231	?XFSTAT Status Flags	2-869
2-232	?XFSTAT Directory and Other Packet File Status	2-871
2-233	?XFSTAT Unit Packet File Status	2-876
2-234	Contents of ?XGTACP Packet	2-884
2-235	Contents of ?XINIT Packet	2-889
2-236	Contents of 16-Bit ?XINIT Packet	2-894
2-237	Contents of ?XPSTAT Standard Parameter Packet	2-905
2-238	Contents of ?XPSTAT Extension Packet for AOS/VS and AOS/VS II	2-908
A-1	Sample Program Sets and their Descriptions	A-1
B-1	SYSLOG Event Codes and Record Lengths	B-3

Figures

Figure	?A Through ?Q	
1-1	Parametric Coding Example	1-3
1-2	Listing File of Program DIRCREATE.SR	1-6
2-1	Structure of ?BLKIO Packet	2-20
2-2	Examples of Read Next Allocated Element Option	2-23
2-3	Structure of Physical I/O Controller Status Block	2-25
2-4	Structure of ?CLASS Packet	2-37
2-5	Structure of ?CLOSE Packet	2-39
2-6	Structure of ?CLSCHEM Packet	2-44
2-7	Structure of ?CLSTAT Main Packet	2-47
2-8	Structure of ?CLSTAT Subpacket	2-49
2-9	Structure of ?CMATRIX Main Packet	2-52
2-10	Structure of ?CMATRIX Subpacket	2-53
2-11	Addresses of ?CMATRIX Main Packet and Its Subpacket Offsets	2-55
2-11.1	Structure of ?CONFIG Main Packet	2-58.1
2-11.2	Structure of the ?CONFIG_GET_CURRENT_ROUTE Function SubPacket	2-58.2
2-11.3	Structure of the ?CONFIG_RESET_MRC_CHANNEL Function SubPacket	2-58.4
2-11.4	Structure of the ?CONFIG_RESET_MRC_CTRLR Function SubPacket	2-58.5
2-11.5	Structure of ?CONINFO Main Packet	2-58.7
2-11.6	Structure of ?CON_TCP_RET_TYPE Return Packet	2-58.10
2-11.7	Structure of ?CON_XNS_RET_TYPE Return Packet	2-58.11
2-11.8	Structure of ?CON_CON_RET_TYPE Return Packet	2-58.12
2-11.9	Structure of ?CON_TNET_RET_TYPE Return Packet	2-58.13
2-11.10	Structure of ?CON_ITC_MIN_DATA Return Packet	2-58.14
2-11.11	Structure of ?CON_TSC_MIN_DATA Return Packet	2-58.15
2-11.12	Structure of ?CON_PVC_RET_TYPE Return Packet	2-58.16
2-11.13	Structure of ?CON_PVC_NAME Return Packet	2-58.19
2-11.14	Structure of ?CON_PVC_NAME_PORT Return Packet	2-58.20
2-11.15	Structure of ?CON_PVC_IP Return Packet	2-58.21
2-11.16	Structure of ?CON_PVC_IP_PORT Return Packet	2-58.22
2-11.17	Structure of ?CON_PVC_ETH Return Packet	2-58.23
2-11.18	Structure of ?CON_PVC_PORT Return Packet	2-58.24
2-11.19	Structure of ?CON_PVC_NET Return Packet	2-58.25
2-12	Structure of ?CPMAX packet	2-59
2-13	Structure of ?CREATE IPC Packet	2-62
2-14	Structure of ?CREATE Time Block	2-63
2-15	Structure of ?CREATE Directory Packet	2-64
2-16	Structure of ?CREATE Packet for Other File Types	2-66
2-17	Structure of ?CRUDA Packet	2-71
2-18	Structure of ?DELETE Packet	2-87
2-19	Extended Task Definition Packet	2-91
2-20	Structure of ?ENBRK Packet	2-96
2-21	Error Code Structure in ERMES File	2-100
2-22	Structure of ?EXEC Packet for Unlabeled Mount Function ?XFMUN	2-106
2-23	Structure of ?EXEC Extended Packet for Unlabeled Mount Function ?XFXUN	2-107

Figure**?A Through ?Q**

2-24	Structure of ?EXEC Packet for Labeled Mount Function ?XFMLT	2-107
2-25	Structure of ?EXEC Extended Packet for Labeled Mount Function ?FXXML	2-108
2-26	Structure of ?EXEC Packet for Dismounting a Tape, ?XFDUN	2-109
2-27	Structure of ?EXEC Packet for Tape Backup	2-110
2-28	Structure of ?EXEC Packet for Queue Requests	2-113
2-29	Structure of the IPC Print Notification Message from ?EXEC	2-123
2-30	Structure of ?XFUSR Subpacket of ?EXEC System Call	2-124
2-31	Structure of AOS/VS ?EXEC Packet for Hold, Unhold, or Cancel Queue Requests.	2-127
2-32	Structure of ?EXEC Packet for Status Information	2-129
2-33	Structure of ?EXEC Packet for Extended Status Information	2-130
2-34	Structure of ?EXEC Packet for a MOUNT Queue Request	2-131
2-35	Structure of ?EXEC Packet for Dismounting a Unit (extended request)	2-132
2-36	Structure of ?EXEC Packet for Changing Queuing Parameters	2-133
2-37	Structure of ?EXEC Packet for Obtaining Queue Names	2-137
2-38	Structure of ?EXEC Packet for Obtaining QDISPLAY Information	2-139
2-39	Structure of ?FEDFUNC Packet to Change Radix	2-145
2-40	Structure of ?FEDFUNC Packet to Open Symbol Table File	2-145
2-41	Structure of ?FEDFUNC Packet to Evaluate a FED String	2-145
2-42	Structure of ?FEDFUNC Packet to Disassemble an Instruction	2-146
2-43	Structure of ?FEDFUNC Packet to Insert a Temporary Symbol	2-146
2-44	Structure of ?FEDFUNC Packet to Delete a Temporary Symbol	2-147
2-45	Structure of Device Control Table (DCT)	2-152
2-46	Structure of Map Definition Table	2-153
2-47	Structure of ?FLOCK Packet	2-160
2-48	Structure of ?FSTAT Packet for Unit Files	2-165
2-49	Structure of ?FSTAT Packet for IPC Files	2-166
2-50	Structure of ?FSTAT Packet for Directory Files	2-167
2-51	Structure of ?FSTAT Packet for Other File Types	2-168
2-52	Structure of ?SSTS Packet	2-169
2-53	Structure of ?FUNLOCK Packet	2-173
2-54	Structure of ?GACL Packet	2-175
2-55	Structure of ?GNFN Packet	2-208
2-56	Structure of ?GOPEN Packet for IPC Files	2-211
2-57	Structure of Standard ?GOPEN Packet	2-212
2-58	Structure of ?GOPEN Packet Extension	2-215
2-59	Structure of ?GPID Packet	2-218
2-60	Structure of ?GPOS Packet	2-221
2-61	Structure of the ?GRAPHICS Main Packet	2-225
2-62	Structure of the ?GRAPH_OPEN_WINDOW_PIXELMAP Subpacket	2-226
2-63	Structure of the ?GRAPHICS_CREATE_MEMORY_PIXELMAP Subpacket	2-228
2-64	Structure of the ?GRAPH_PIXELMAP_STATUS Subpacket	2-229
2-65	Structure of the ?GRAPH_SET_CLIP_RECTANGLE Subpacket	2-231
2-66	Structure of the ?GRAPH_MAP_PIXELMAP Subpacket	2-233
2-67	Structure of the ?GRAPH_WRITE_PALETTE Subpacket	2-235
2-68	Structure of the ?GRAPH_READ_PALETTE Subpacket	2-236
2-69	Structure of the ?GRAPHICS_SET_DRAW_ORIGIN Subpacket	2-237
2-70	Structure of the ?GRAPHICS_GET_DRAW_ORIGIN Subpacket	2-238
2-71	Structure of ?GROUP Packet	2-241
2-72	Structure of ?GROUP log entry	2-241

Figure**?A Through ?Q**

2-73	Structure of ?GTIME Packet	2-248
2-74	Structure of ?GTMES Packet	2-251
2-75	Structure of ?GTRUNCATE Packet	2-260
2-76	Structure of ?GUHPI Packet	2-264
2-77	Structure of Device Control Table (DCT) for 32-Bit Processes	2-271
2-78	Structure of Device Control Table (DCT) for 16-Bit Processes	2-272
2-78.1	Structure of Extended Packet for 16-Bit Processes	2-272
2-79	Structure of Map Definition Table	2-273
2-80	Structure of ?IHIST Packet	2-287
2-81	Structure of ?IREC Header	2-296
2-82	Structure of Offset ?IUFL	2-298
2-82.1	Structure of Termination Message from a 32-bit B- or C-Type Process	2-304.1
2-82.2	Structure of Termination Message from a 16-bit B- or C-Type Process	2-304.4
2-83	Structure of ?ISEND Header	2-306
2-84	Structure of ?IS.R Header	2-310
2-85	Structure of ?JPINIT Packet	2-318
2-86	Structure of ?JPMOV Packet	2-321
2-87	Structure of ?JPREL Packet	2-323
2-88	Structure of ?JPSTAT Main Packet	2-325
2-89	Structure of ?JPSTAT General Information Subpacket	2-326
2-90	Structure of ?JPSTAT Specific Information Subpacket	2-326
2-91	Structure of ?LABEL Packet	2-337
2-92	Structure of ?LDUINFO Main Packet	2-341
2-93	Structure of ?LDU_PKT Subpacket	2-343
2-94	Structure of ?LDU_PKT Array Record	2-346
2-95	Structure of ?PIECE_PKT Subpacket	2-347
2-96	Structure of ?LOCALITY Packet	2-355
2-97	Structure of ?LOGEV Event Logging Format	2-361
2-97.1	Structure of Termination Message from a 32-bit B- or C-Type Process	2-304.1
2-97.2	Structure of Termination Message from a 16-bit B- or C-Type Process	2-304.4
2-98	Structure of ?LPCLASS Packet	2-363
2-99	Structure of ?LPCREA Packet	2-366
2-100	Structure of ?LPDELE Packet	2-368
2-101	Structure of ?LPSTAT Main Packet	2-370
2-102	Structure of ?LPSTAT General Information Subpacket	2-371
2-103	Structure of ?LPSTAT Specific Information Subpacket	2-371
2-104	Structure of ?MAPDV Packet	2-375
2-105	Structure of ?MBFC Packet	2-378
2-106	Structure of ?MBTC Packet	2-380
2-107	Structure of ?MIRROR Packet	2-387
2-108	Structure of ?MIRROR Subpacket	2-390
2-109	Structure of 16-Bit ?MIRROR Packet	2-390
2-110	Structure of 16-Bit ?MIRROR Subpacket	2-393
2-111	Structure of ?MPHIST Packet	2-396
2-112	Structure of ?NTIME Packet	2-401
2-113	Structure of ?OPEN Packet	2-407
2-114	Sample Delimiter Table	2-416
2-115	Structure of ?OPEN Extension Packet for Pipes	2-416
2-116	Structure of Labeled Magnetic Tape Packet Extension	2-419

Figure**?A Through ?Q**

2-117	Structure of ?OPER Main Packet	2-425
2-118	Structure of ?OPON Subpacket	2-427
2-119	Structure of ?OPOFF Subpacket	2-429
2-120	Structure of ?OPSEND Subpacket	2-432
2-121	Structure of ?OPRCV Subpacket	2-432
2-122	Structure of ?OPRESP Subpacket	2-434
2-123	Structure of ?OPINFO Subpacket	2-435
2-124	Structure of ?OPEX Main Packet	2-438
2-125	Structure of Access Command Subpacket	2-442
2-126	Structure of Align Command Subpacket	2-443
2-127	Structure of Batch_List Command Subpacket	2-444
2-128	Structure of Batch_Output Command Subpacket	2-445
2-129	Structure of Binary Command Subpacket	2-446
2-130	Structure of Brief Command Subpacket	2-447
2-131	Structure of Cancel Command Subpacket	2-448
2-132	Structure of Consolestatus Command Subpacket	2-449
2-133	Structure of Continue Command Subpacket	2-451
2-134	Structure of CPL Command Subpacket	2-452
2-135	Structure of Create Command Subpacket	2-453
2-136	Structure of Defaultforms Command Subpacket	2-454
2-137	Structure of Disable Command Subpacket	2-455
2-138	Structure of Dismounted Command Subpacket	2-456
2-139	Structure of Elongate Command Subpacket	2-457
2-140	Structure of Enable Command Subpacket	2-458
2-141	Structure of Even Command Subpacket	2-460
2-142	Structure of Flush Command Subpacket	2-461
2-143	Structure of Forms Command Subpacket	2-462
2-144	Structure of Halt Command Subpacket	2-463
2-145	Structure of Headers Command Subpacket	2-464
2-146	Structure of Hold Command Subpacket	2-465
2-147	Structure of Limit Command Subpacket	2-466
2-148	Structure of Logging Command Subpacket	2-467
2-149	Structure of LPP Command Subpacket	2-469
2-150	Structure of Mapper Command Subpacket	2-470
2-151	Structure of Mounted Command Subpacket	2-472
2-152	Structure of Mountstatus Command Subpacket	2-473
2-153	Structure of Operator Command Subpacket	2-476
2-154	Structure of Pause Command Subpacket	2-477
2-155	Structure of Premount Command Subpacket	2-478
2-156	Structure of Priority Command Subpacket	2-479
2-157	Structure of Prompts Command Subpacket	2-480
2-158	Structure of Qpriority Command Subpacket	2-481
2-159	Structure of Refused Command Subpacket	2-483
2-160	Structure of Release Command Subpacket	2-484
2-161	Structure of Restart Command Subpacket	2-485
2-162	Structure of Silence Command Subpacket	2-486
2-163	Structure of Spoolstatus Command Subpacket	2-487
2-164	Structure of Stack Command Subpacket	2-491
2-165	Structure of Start Command Subpacket	2-492

Figure**?A Through ?Q**

2-166	Structure of Status Command Subpacket	2-495
2-167	Structure of Stop Command Subpacket	2-499
2-168	Structure of Trailers Command Subpacket	2-500
2-169	Structure of Unhold Command Subpacket	2-501
2-170	Structure of Unitstatus Command Subpacket	2-502
2-171	Structure of Unlimit Command Subpacket	2-504
2-172	Structure of Unsilence Command Subpacket	2-505
2-173	Structure of User-Command Subpacket	2-506
2-174	Structure of Verbose Command Subpacket	2-508
2-175	Structure of ?PCLASS Packet	2-515
2-176	Structure of ?PIDS Packet	2-518
2-177	Structure of ?PMTPF Packet	2-520
2-178	Structure of ?PRDB/?PWRB Packet	2-526
2-179	Structure of ?PROC Packet	2-537
2-180	Structure of ?PROC Extension Packet for AOS/VS and AOS/RT32	2-546
2-181	Structure of ?PROC Extension Packet for AOS/VS II	2-546
2-182	Structure of ?PROFILE Parameter Packet	2-552
2-183	Structure of ?PROFILE Field Descriptor Packet	2-553
2-184	Structure of ?PSTAT Memory Descriptor	2-561
2-185	Structure of ?PSTAT Packet	2-562
2-186	Structure of the ?PTRDEVICE Main Packet	2-569
2-187	Tablet States	2-572
2-188	An Example of a Tablet Area Menu	2-573
2-189	Structure of the ?PTRDEV_SET_EVENTS Subpacket	2-573
2-190	Structure of the ?PTRDEV_SET_DELTA Subpacket	2-576
2-191	Structure of the ?PTRDEV_LAST_EVENT Subpacket	2-577
2-192	Subpacket for ?PTRDEV_SET_POINTER Subpacket	2-579
2-193	Structure of the ?PTRDEV_GET_PTR_STATUS Subpacket	2-581
2-194	Structure of the ?PTRDEV_GENERATE_EVENT Subpacket	2-585
2-195	Structure of the ?PTRDEV_GET_PTR_LOCATION Subpacket	2-587
2-196	Structure of the ?PTRDEV_GET_TABLET_LOCATION Subpacket	2-589
2-197	Structure of ?PWDCRYP Packet	2-591

?R Through ?Z

2-198	Structure of ?RDB/?WRB Packet	2-597
2-199	Structure of ?RDUDA/?WRUDA Packet	2-603
2-200	Structure of ?READ/?WRITE Packet	2-605
2-201	Structure of Screen Management Packet Extension	2-606
2-202	Structure of Selected Field Translation Packet Extension	2-606
2-203	Structure of New Screen Management Packet Extension	2-606
2-204	Structure of Screen-Management Packet Extension	2-612
2-205	Structure of Selected Field Translation Packet Extension	2-617
2-206	Selected Field Translation Packet Sample Listing	2-619
2-207	Structure of ?RENAME Packet	2-632
2-208	Structure of ?RNGPR Packet	2-641
2-209	Structure of ?RTODC Packet	2-646
2-210	Structure of ?RUNTM Packet	2-649
2-211	Structure of ?SACL Packet	2-652
2-212	Structure of ?SATR Packet	2-654

Figure**?R Through ?Z**

2-213	Polling List Defined by a Control Station	2-666
2-214	Poll and Select Addresses Defined by a Tributary	2-666
2-215	Structure of ?SEBL Packet	2-670
2-216	Station Identification System Call Sequence	2-673
2-217	Structure of the ?SERMSG Packet	2-684
2-218	Structure of ?SGES Packet	2-686
2-219	Structure of ?SINFO Packet	2-693
2-220	Structure of ?SONS Packet	2-697
2-221	Structure of ?SOPPF Packet	2-702
2-222	Structure of ?SPAGE Packet	2-705
2-223	Structure of ?SRCV Packet	2-710
2-224	ITB Receive Buffer Format	2-717
2-225	Structure of ?SSND Packet	2-721
2-226	Structure of ?SYLOG Exclusion Bit Map Packet	2-743
2-227	Structure of ?SYSPRV Packet	2-745
2-228	Structure of Standard Task Definition Packet	2-748
2-229	Stack Parameters for Initiating One or More Tasks	2-750
2-230	Extended Task Definition Packet	2-751
2-231	Structure of Termination Message a 32-bit Process Receives	2-761
2-232	Structure of Termination Message a 16-bit Process Receives	2-764
2-233	Structure of ?UIDSTAT Packet	2-777
2-234	Structure of ?VMEM Packet	2-788
2-235	Structure of ?VTFCREATE Packet	2-791
2-236	Execution of Two Virtual Timers	2-796
2-237	Structure of ?VTFKILL Packet	2-798
2-238	Structure of ?VTFMODIFY Packet	2-800
2-239	Structure of ?WHIST Packet	2-811
2-240	Sample Histogram Parameters	2-811
2-241	Structure of the ?WINDOW Main Packet	2-816
2-242	Structure of the ?WIN_CREATE_WINDOW Subpacket	2-818
2-243	Structure of the ?WIN_DEFINE_PORTS Subpacket	2-823
2-244	Structure of the ?WIN_SET_USER_INTERFACE Subpacket	2-828
2-245	Structure of the ?WIN_GET_USER_INTERFACE Subpacket	2-830
2-246	Structure of the ?WIN_GET_TITLE Subpacket	2-831
2-247	Structure of the ?WIN_GTITLE Subpacket	2-832
2-248	Structure of the ?WIN_WINDOW_STATUS Subpacket	2-834
2-249	Structure of the ?WIN_DEVICE_STATUS Subpacket	2-838
2-250	Structure of the ?WIN_RETURN_GROUP_WINDOWS Subpacket	2-840
2-251	Structure of the ?WIN_RETURN_DEVICE_WINDOWS Subpacket	2-841
2-252	Structure of ?XCREATE Main Packet	2-850
2-253	Structure of ?XCREATE Time-Block Subpacket	2-853
2-254	Structure of ?XCREATE Other Subpacket	2-854
2-255	Structure of ?XCREATE Directory Subpacket	2-855
2-256	Structure of ?XCREATE IPC Subpacket	2-856
2-257	Structure of ?XCREATE Link Subpacket	2-857
2-258	Example of ?XCREATE Main Packet for TXT File	2-858
2-259	Example of ?XCREATE Other Subpacket for TXT File	2-858
2-260	Example of ?XCREATE Main Packet for DIR File	2-859
2-261	Example of ?XCREATE Time Subpacket for DIR File	2-859

Figure**?R Through ?Z**

2-262	Example of ?XCREATE Directory Subpacket for DIR File	2-860
2-263	Example of ?XCREATE Main Packet for IPC File	2-860
2-264	Example of ?XCREATE IPC Subpacket for IPC File	2-861
2-265	Example of ?XCREATE Main Packet for LNK File	2-861
2-266	Example of ?XCREATE Link Subpacket for LNK File	2-861
2-267	Structure of ?XFSTAT IPC Packet	2-866
2-268	Structure of ?XFSTAT Directory Packet	2-870
2-269	Structure of ?XFSTAT Other Packet	2-873
2-270	Structure of ?XFSTAT Unit Packet	2-875
2-271	Example of Pathname Supplied	2-879
2-272	Example of Directory Type Grouping	2-881
2-273	Structure of ?XGTACP Packet	2-883
2-274	Structure of ?XINIT Packet	2-888
2-275	Structure of ?XINIT Subpacket	2-892
2-276	Structure of 16-Bit ?XINIT Packet	2-893
2-277	Structure of 16-Bit ?XINIT Subpacket	2-897
2-278	Structure of ?XPSTAT Packet	2-902
2-279	Structure of ?XPSTAT Standard Memory Descriptor	2-904
2-280	Structure of ?XPSTAT Extended Memory Descriptor (AOS/VS and AOS/VS II)	2-904
2-281	Structure of ?XPSTAT Extension Packet for AOS/VS and AOS/VS II	2-909
A-1	Listing of Program HEAR.SR	A-3
A-2	Listing of Program SPEAK.SR	A-7
A-3	Listing of Program SON.SR	A-9
A-4	Listing of Program RUNTIME.SR	A-11
A-5	Listing of Program RINGLOAD.SR	A-14
A-6	Listing of Program INRING.SR	A-16
A-7	Listing of Program GATE.ARRAY.SR	A-17
A-8	Listing of Program FILCREATE.SR	A-19
A-9	Listing of Program WRITE.SR	A-22
A-10	Listing of Program DLIST.SR	A-26
A-11	Listing of Program NEWTASK.SR	A-29
A-12	Listing of Program BOOMER.SR	A-32
A-13	Listing of Program TIMEOUT.SR	A-37
A-14	Listing of Program DIRCREATE.F77	A-41
A-15	File DIRCREATE_SYMBOLS	A-41
A-16	File DIRCREATE_SYMBOLS.F77.IN	A-42
A-17	Listing of Subroutine CHECK.F77	A-42
A-18	File CHECK_SYMBOLS	A-43
A-19	File CHECK_SYMBOLS.F77.IN	A-43
A-20	Listing of Program CREATE_WINDOW.SR	A-46
A-21	Code to Turn Permanence On in Program CREATE_WINDOW	A-55
A-22	Code to Turn Permanence Off in Program CREATE_WINDOW	A-55
A-23	Listing of Program GRAPHICS_SAMPLE.SR	A-57
A-24	Possible Results of Executing Program GRAPHICS_SAMPLE	A-76
B-1	Log Record Header	B-2
B-2	Log Record Codes, Events, and Message Lengths, Excluding Header	B-12
B-3	An Octal and Decimal DISPLAY of a System Log File	B-20
B-4	Octal and Decimal Versions of a SYSLOG Record	B-21

Table 2-1. Summary of AOS/VS and AOS/RT32 System Calls

Function: Connection Management		AOS/	AOS/	AOS/	OS
System Call	Description	VS	VSII	RT32	Diff
?CON	Becomes a customer of a specified server.	Y	Y	Y	
?CTERM	Terminates a customer process.	Y	Y	Y	
?DCON	Breaks a connection (disconnects) in Ring 7.	Y	Y	Y	
?DRCON	Breaks a connection (disconnects).	Y	Y	Y	
?MBFC	Moves bytes from a customer's buffer.	Y	Y	Y	
?MBTC	Moves bytes to a customer's buffer.	Y	Y	Y	
?PCNX	Passes a connection from one server to another in Ring 7.	Y	Y	Y	
?PRCNX	Passes a connection from one server to another.	Y	Y	Y	
?RESIGN	Resigns as a server.	Y	Y	Y	
?SERVE	Becomes a server.	Y	Y	Y	
?SIGNL	Signals another task.	Y	Y	Y	
?SIGWT	Signals another task, and then waits for a signal.	Y	Y	Y	
?VCUST	Verifies a customer in Ring 7.	Y	Y	Y	
?VRCUST	Verifies a customer in a specified ring.	Y	Y	Y	
?WTSIG	Waits for a signal from another task or process.	Y	Y	Y	
Function: Multiprocessor Management		AOS/	AOS/	AOS/	OS
System Call	Description	VS	VSII	RT32	Diff
?JPINIT	Initializes a job processor.	Y	Y		
?JPMOV	Moves a job processor to a new logical processor.	Y	Y		
?JPREL	Releases a job processor.	Y	Y		
?JPSTAT	Gets the status of a job processor.	Y	Y		

(continued)

Table 2-1. Summary of AOS/VS and AOS/RT32 System Calls

Function: Class Scheduling		AOS/VS	AOS/VSII	AOS/RT32	OS Diff
System Call	Description				
?CLASS	Gets or sets class IDs.	Y	Y		
?CLSCHEM	Enables, disables, or examines class scheduling.	Y	Y		
?CLSTAT	Returns class scheduling statistics.	Y	Y		
?CMATRIX	Gets or sets the class matrix.	Y	Y		
?LPCLASS	Gets/sets logical processor class assignments.	Y	Y		
?LPCREA	Creates a logical processor.	Y	Y		
?LPDELE	Deletes a logical processor.	Y	Y		
?LPSTAT	Gets the status of a logical processor.	Y	Y		
Function: AOS/VS System Resources		AOS/VS	AOS/VSII	AOS/RT32	OS Diff
System Call	Description				
?BNAME	Determines whether process name/queue name is on local or remote host.	Y	Y	Y	
?CDAY	Converts a scalar date value.	Y	Y	Y	
?CONFIG	Displays or resets the current message-based reliable (MRC) device routes.		Y		
?CONINFO	Requests addressing information on a terminal or a console.		Y		
?CTOD	Converts a scalar time value.	Y	Y	Y	
?ENQUE	Sends a message to IPC and spooler files.	Y	Y	Y	
?ERMSG	Reads the error message file.	Y	Y	Y	
?EXEC	Requests a service from EXEC.	Y	Y		
?FDAY	Converts date to a scalar value.	Y	Y	Y	
?FEDFUNC	Interfaces to File Editor utility.	Y	Y	Y	
?FTOD	Converts time of day to a scalar value.	Y	Y	Y	
?GDAY	Gets the current date.	Y	Y	Y	
?GHRZ	Gets the frequency of the system clock.	Y	Y	Y	
?GSID	Gets the system identifier.	Y	Y	Y	
?GTIME	Gets the time, date, and time zone.	Y	Y		
?GTMES	Gets an initial IPC message.	Y	Y	Y	
?GTOD	Gets the time of day.	Y	Y	Y	
?GUHPI	Gets unique hardware processor ID.	Y	Y		
?ITIME	Returns the OS-format internal time.	Y	Y	Y	
?LOGCALLS	Logs system calls.	Y	Y	Y	
?LOGEV	Enters an event in the system log file.	Y	Y		
?NTIME	Sets the time, date, and time zone.	Y	Y		
?OPER	Creates and maintains an operator interface.	Y	Y		
?OPEX	Communicates between the current process and an operator process.	Y	Y		
?PWDCRYP	Performs a password data encryption request.	Y	Y		
?RTODC	Reads the time-of-day conversion data.	Y	Y		

(continued)

Table 2-1. Summary of AOS/VS and AOS/RT32 System Calls

Function: AOS/VS System Resources		AOS/VS	AOS/VSII	AOS/RT32	OS Diff
System Call	Description				
?SDAY	Sets the system calendar.	Y	Y	Y	
?SINFO	Gets selected information about the current operating system.	Y	Y	Y	Y
?SSID	Sets the system identifier.	Y	Y	Y	
?STOD	Sets the system clock.	Y	Y	Y	
Function: User Devices		AOS/VS	AOS/VSII	AOS/RT32	OS Diff
System Call	Description				
?CLRDV	Clears a device.	Y	Y	Y	
?DDIS	Disables access to all devices.	Y	Y	Y	
?DEBL	Enables access to all devices.	Y	Y	Y	
?IDEF	Defines a user device.	Y	Y	Y	
?IMSG	Receives an interrupt service message.	Y	Y	Y	
?IRMV	Removes a user device.	Y	Y	Y	
?IXIT	Exits from an interrupt service routine.	Y	Y	Y	
?IXMT	Transmits a message from an interrupt service routine.	Y	Y	Y	
?LEFD	Disables LEF mode.	Y	Y	Y	
?LEFE	Enables LEF mode.	Y	Y	Y	
?LEFS	Returns the current LEF mode status.	Y	Y	Y	
?MAPDV	Maps a device into logical address space.	Y	Y	Y	Y
?STMAP	Sets the data channel map.	Y	Y	Y	
Function: Bisynchronous Communications		AOS/VS	AOS/VSII	AOS/RT32	OS Diff
System Call	Description				
?SDBL	Disables a BSC line.	Y	Y	Y	
?SDPOL	Defines a polling list or a poll-address/select-address pair.	Y	Y	Y	
?SDRT/ ?SERT	Disables/re-enables a relative terminal.	Y	Y	Y	
?SEBL	Enables a BSC line.	Y	Y	Y	
?SERT	Re-enables a relative terminal.	Y	Y	Y	
?SGES	Gets BSC error statistics.	Y	Y	Y	
?SRCV	Receives data or a control sequence over a BSC line.	Y	Y	Y	
?SSND	Sends data or a control sequence over a BSC line.	Y	Y	Y	

(continued)

Table 2-1. Summary of AOS/VS and AOS/RT32 System Calls

Function: 16-bit Processes		AOS/	AOS/	AOS/	OS
System Call	Description	VS	VSII	RT32	Diff
?DELAY	Suspends a 16-bit task for a specified interval.	Y	Y	Y	
?GCRB	Gets the base of the current resource.	Y	Y	Y	
?IDSTAT	Returns task status word.	Y	Y	Y	
?IESS	Initializes an extended state save area.	Y	Y	Y	
?IHIST	Starts a histogram for a 16-bit process.	Y	Y	Y	
?KCALL	Keeps the calling resource and acquires a new resource.	Y	Y	Y	
?OVEX	Releases an overlay and returns.	Y	Y	Y	
?OVKIL	Exits from an overlay and kills the calling task.	Y	Y	Y	
?OVL0D	Loads and goes to an overlay.	Y	Y	Y	
?OVREL	Releases an overlay area.	Y	Y	Y	
?RCALL	Releases one resource and acquires a new one.	Y	Y	Y	
?RCHAIN	Chains to a new procedure.	Y	Y	Y	
?SERMSG	Returns text for associated error code.	Y	Y	Y	
?WALKBACK	Returns information about previous frames in the stack.	Y	Y	Y	

(concluded)

?ALLOCATE

Allocates disk blocks.

?ALLOCATE ?RDB/?WRB packet address

error return

normal return

Input

AC0 Reserved (Set to 0.)

AC1 Channel number

AC2 Address of the ?RDB/?WRB

Output

AC0 Undefined

AC1 Number of bytes allocated
(number of blocks * 512)

AC2 Unchanged
packet

Error Codes in AC0

ERCPD Control point directory maximum exceeded

ERFNO Channel not open

ERICB Insufficient contiguous disk blocks

ERSIM Simultaneous requests on same channel

ERVWP Invalid word pointer passed as a system call argument

ERWAD Write access denied

ER_FS_DIRECTORY_NOT_AVAILABLE

Directory not available because the LDU was force released (AOS/VS II only)

ER_FS_TLA_MODIFY_VIOLATION

Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet

File system error codes

Why Use It?

?ALLOCATE makes sure that subsequent I/O will not cause a calling process to exceed its control point directory's maximums.

Who Can Use It?

There are no special process privileges needed to issue this call. You must have obtained a channel number to the file, via ?GOPEN or ?SOPEN, before issuing ?ALLOCATE. Also, you must have had Write access to the file at the time of the ?GOPEN or ?SOPEN call.

What It Does

?ALLOCATE allocates disk blocks for specified data elements and initializes to zero those elements that do not actually exist. If the data elements already exist, the contents do not change and ?ALLOCATE takes the normal return. ?ALLOCATE pends only the task within the process that issues ?ALLOCATE.

The operating system determines the actual number of data elements to allocate from the information that u supply in the following ?RDB/?WRB packet offsets:

?PSTI Right byte: number of data blocks to allocate
Left byte: ignored

?PRNH Starting data block number (doubleword)

?ALLOCATE Continued

The operating system ignores all other offsets in the ?RDB/?WRB packet.

Note that to use ?ALLOCATE, the file must have been opened with ?GOPEN or ?SOPEN, not with ?OPEN.

Notes

- See the descriptions of ?GOPEN, ?SOPEN, and ?RDB/?WRB in this chapter.

?ASSIGN

Assigns a character device to a process.

AOS/VS

?ASSIGN

error return

normal return

Input

AC0 Byte pointer to the name
of the device to assign

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Undefined

AC2 Undefined

Error Codes in AC0

ERASS Assign error — already your device
ERIDT Illegal device type
ERIFL IAC (Intelligent Asynchronous Controller) failure
ERVBP Invalid byte pointer passed as a system ll argument

Why Use It?

?ASSIGN allows you to link a character device exclusively to your process until you issue ?DEASSIGN or your process terminates. You can issue ?ASSIGN against a device if you want to close it periodically without the risk of losing it to another process.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?ASSIGN assigns the character device that you specify in AC0 to the calling process. When you assign a device to a process, you are reserving the device for the exclusive use of that process. Once you assign a device to a process, it remains the exclusive property of that process until the process terminates or breaks the assignment by issuing ?DEASSIGN. You cannot assign a device that is currently assigned to another process (such as a device enabled for spooling.)

Notes

- See the description of ?DEASSIGN in this chapter.
- See the descriptions of ?OPEN and ?CLOSE in this chapter for information on implicit device assignments and deassignments.

?AWIRE

Changes the wiring characteristics of the Agent.

AOS/VS

?AWIRE

error return

normal return

Input

AC0 Not used (Set to 0.)

AC1 Set ?AWENT to wire the entire Agent;
Set ?AWUDS to wire only the areas of the Agent required to support user devices

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERPTY Illegal process type

ERPRE Invalid parameter passed as system call argument

Why Use It?

Use the ?AWIRE system call to unwire Agent pages if your process is resident for the purpose of supporting user devices. This will free up several pages of physical memory which can be used for other purposes. While this may seem to degrade efficiency of a resident process, it actually increases the efficiency of the system as a whole.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

This call allows a resident process to change the wiring characteristics of the Agent portion of the operating system. Processes that ?PROC declares resident and processes that do a ?CTYPE against themselves to become resident will, by default, have their entire Agent wired. This is done to decrease the number of page faults taken by resident processes, thereby increasing their efficiency. However, resident processes that wish to support user-defined devices, and do not need the increased efficiency of a wired Agent can decrease their working set size by issuing the ?AWIRE system call with ?AWUDS set in AC0. This will cause the Agent to unwire all pages, except those required for user device support. This will also free up several pages of physical memory so they can be used for other purposes. If, after issuing the ?AWIRE system call with ?AWUDS set in AC0, the process would like to wire the entire Agent again to increase efficiency, the process can reissue the ?AWIRE system call with ?AWENT set in AC0. This will cause the Agent to wire its entire address space. If multiple ?AWIRE system calls are issued with the same option, they will take the normal return, but will have no effect.

?BLKIO

Performs (reads/writes) block I/O.

?BLKIO [*packet address*]

error return

normal return

Operating System Differences

AOS/RT32 does not support modified sector I/O.

Input

AC0 Reserved (Set to 0.)
AC1 Reserved (Set to 0.)
AC2 Address of the ?BLKIO packet unless you pass the address as an argument to the system call.

Output

AC0 Undefined
AC1 Undefined
AC2 Address of the packet.

Error Codes in AC0

ER32U Record too large for this device when using ?BM32R for a 32-bit record number
ERDIO Attempt to issue MCA direct I/O with outstanding requests
EREOF End of file
ERVWP Invalid word pointer passed as system call argument
ERPUF Physical unit failure
ERFAD File access denied
ERRAD Read access denied
ERSPC File space exhausted
ERWAD Write access denied
ERIoT Wrong type I/O for open type
ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)
ER_FS_TLA_MODIFY_VIOLATION
Attempt to modify file with ?ODTL value supplied in ?GOPEN packet (write block only)

Why Use It?

Use ?BLKIO for block reads or writes, or for physical block reads or writes (see ?RDB/?WRB and ?PRDB/?PRWB.) ?BLKIO is particularly useful for its read next allocated element feature when you're reading files with many unallocated elements.

Who Can Use It?

There are no special process privileges needed to issue this call. You must have Read access to the file that you want to read from and Write access to the file that you want to write to.

What It Does

?BLKIO performs block I/O on a file. This system call combines the functionality of the ?RDB/?WRB (block I/O) and ?PRDB/?PWRB (physical block I/O) system calls, and includes some additional functionality. When you use ?BLKIO for a block read, a block write, a physical block read, or a physical block write, the system call acts EXACTLY like the ?RDB/?WRB and ?PRDB/?PWRB system calls. (For details, refer to the sections on these system calls in this chapter.)

?BLKIO Continued

In addition, ?BLKIO allows you to read the next allocated element in a disk file skipping unallocated blocks. Writing to any disk block in an element allocates the element to be read. Note that the read next allocated element option can only be used if the call is for a read and physical I/O has not been selected.

The ability to read the next allocated element is especially useful if you're reading a large file with many unallocated elements; ?BLKIO skips over these elements and reads only from the allocated ones. As a result, if you read a long file using this option, you save time. (?RDB reads every element, allocated or not.) You can see some illustrations of this feature later in the discussion of ?BLKIO.

Before you can issue ?BLKIO against a file, you must open the file with the ?GOPEN system call. You must also set up a parameter packet of length ?BLTH in your address space. Figure 2-1 shows the structure of the packet, and Table 2-2 describes its contents.

Under AOS/VS only, ?BLKIO also allows you to perform modified sector I/O on disks that support this hardware feature. Modified sector I/O must be used in conjunction with physical block I/O. This type of I/O allows you to read and to clear only the modified blocks of a disk unit. However, modified sector I/O does not function under AOS/VS II.

	0	7 8	15 16	31
?BPVB	Reserved (must be 0)		Bit status flags	?BSTS
?BCHN	Channel number		Reserved (set to 0)	?BERR
?BADR	Data buffer address			?BADL
?BBLN	Block number of first block to be transferred			?BBLL
?BBLC	Reserved (must be 0)	No. blocks to transfer	Last block byte count	?BLBB
?BTBC	Total number of bytes read or written			?BTBL
?BBAN	First disk block actually transferred (read next allocated element only)			?BBAL
?BBAC	Number of disk blocks transferred (read next allocated element only)		Address of physical I/O controller status block -- high order bits	?BPER
?BPEL	Address of physical I/O controller status block -- low order bits		Reserved (must be 0)	
?B32N	32-bit record number for tape			
	Reserved (must be 0)		Reserved (must be 0)	
	?BLTH = packet length			

Figure 2-1. Structure of ?BLKIO Packet

When the operating system blocks a process, it also suspends all tasks in the process until a ?UBLPR system call is issued against that process. Suspension on that event is independent of — and has no effect on — the state of any other suspensions that may be in force against a task.

Notes

- See the description of ?UBLPR in this chapter.

?BNAME

Determines whether process name/queue name is on local or remote host.

?BNAME

error return

normal return

Input

AC0 Byte pointer to the process name/queue name

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 One of the following:

- 0 if process name/queue name is on local host
- Bits 17 through 31 contain host ID if process name/queue name is on remote host
- -1 if reference is remote and host has zero host ID

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERPNUM Illegal process name
The input process name/queue name does not contain a colon (:).

Why Use It?

You can use ?BNAME to find out if a particular process or queue is on a local host or on a remote host.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?BNAME determines whether a particular process or queue is on a local or a remote host. To use ?BNAME, you must specify a process name or queue name in AC0. Then, the operating system examines the name to decide whether the process/queue is on a local or a remote host. If the host is remote, or if the process name or the queue name contains the local host name, the operating system returns the host ID in Bits 17 through 31 of AC0.

If the host ID does not exist in directory :NET, ?BNAME assumes the case of a local host.

Who Can Use It?

You need no special process privileges to issue this call. There are no restrictions concerning file access. However, you must connect to a valid server process. After the connection, the server process has access to the memory in the ring from which you issued the call. So you should use ?CON to connect cooperating processes.

What It Does

?CON defines the calling process as a customer of an existing server process that you specify in AC0, and it directs the operating system to create a corresponding connection-table entry. The connection takes place between the caller's PID/ring and a specified server's PID/ring pair.

If the target process is not a declared server (it has not already issued ?SERVE), ?CON fails on error code ERNAS. If the caller tries to connect with itself, ?CON fails and returns error code ERCCS.

If the connection already exists, ?CON resets the status according to the bit mask ?MCOBIT. However, if the customer is trying to reconnect an already broken connection, but the server has not broken it, ?CON fails on error code ERCBK.

When a server disconnects, the operating system sends (by default) an obituary message to the customer. To suppress this message, set ?MCOBIT in AC1.

You cannot connect segment images within the same process.

Notes

- See the description of ?DCON in this chapter.

?CONFIG

Display or reset current MRC routes.

AOS/VS II

?CONFIG [*packet address*]

error return

normal return

Input

AC0 Reserved (set to 0)
AC1 Reserved (set to 0)
AC2 Address of the ?CONFIG packet, unless specified as an argument to ?CONFIG

Output

AC0 Unchanged or an error code
AC1 Unchanged
AC2 Address of the ?CONFIG packet

Error codes returned in AC0

ERPRV Caller not privileged for this action
ERRVN Reserved value not zero
ERICD Illegal function code
ERPKT Illegal packet id
ERMPR System call parameter address error
ERPRE Invalid system call parameter
ERVWP Invalid address passed as system call argument
ERUNC Device unknown to host
ERIDT Illegal device name type

Why Use It?

Use ?CONFIG to either determine the current Message-Based Reliable (MRC) device routes on a system, or to return diverted MRC devices to their primary routes on a specified MRC channel or controller. For MRC devices, primary routes are user-defined and edited in VSGEN, but secondary routes are solely defined by VSGEN and are not user-selectable. A diverted MRC device is one using a secondary route.

Who Can Use it?

Any user who has the access devices privilege.

What It Does

The ?CONFIG system call consists of a main packet and three subpacket functions. The first subpacket function obtains the current route information for a specified MRC device. The other two subpacket functions reset the routes of diverted devices on either a designated channel, or on a controller.

The ?CONFIG_GET CURRENT_ROUTE function returns the current route, the primary route, and the device reset-pending status for a valid MRC device name. The ?CONFIG_RESET_MRC_CHAN function resets the current working routes *back* to their primary routes, if these are available, for all the diverted MRC devices on the designated channel. The ?CONFIG_RESET_MRC_CONTROLLER function resets the current working routes *back* to their primary routes, if these are available, for all the diverted MRC devices on the designated controller.

The ?CONFIG call does not change device routing from primary to secondary routes. Figure 2-11.1 shows the structure of the ?CONFIG main packet.

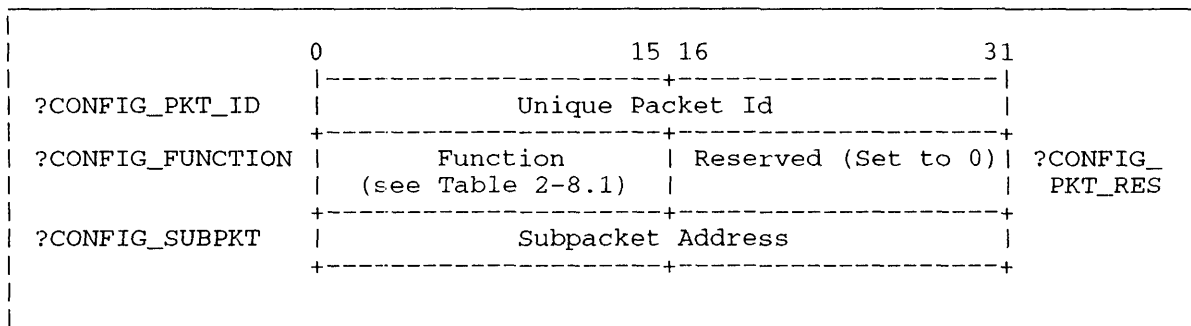


Figure 2-11.1. Structure of ?CONFIG Main Packet

The valid function codes for offset ?CONFIG_FUNCTION are described in Table 2-8.1.

Table 2-8.1. Valid ?CONFIG_FUNCTION Function Codes

Function Code	What It Lets You Do
?CONFIG_GET_CURRENT_ROUTE	Specifies the MRC unit name for the requested route information. The current route, primary route, the unit number, and the reset pending flags are all returned.
?CONFIG_RESET_MRC_CHAN	Specifies the MRC channel's device code for the primary channel requiring resetting. This function attempts to reset all the diverted MRC devices on this channel.
?CONFIG_RESET_MRC_CTRLR	Specifies the MRC channel's device code, and the node number of the primary controller requiring resetting. The channel's device code is the 3-bit IOC, 6-bit device code combination, in which bits 7-9 of the single word contain the IOC number, and bits 10-15 contain the device code. This function attempts to reset all the diverted MRC devices on this controller.

?CONFIG Continued

What ?CONFIG_GET_CURRENT_ROUTE Does

The ?CONFIG_GET_CURRENT_ROUTE function accepts as input a byte pointer to the MRC unit name string. ?CONFIG_GET_CURRENT_ROUTE returns the currently active route components, the primary route components, the unit number, and a flag word. The caller must determine whether or not the routes are diverted and require resetting.

The ?CONFIG_GET_CURRENT_ROUTE subpacket structure is shown in Figure 2-11.2, and its contents are described in Table 2-8.2.

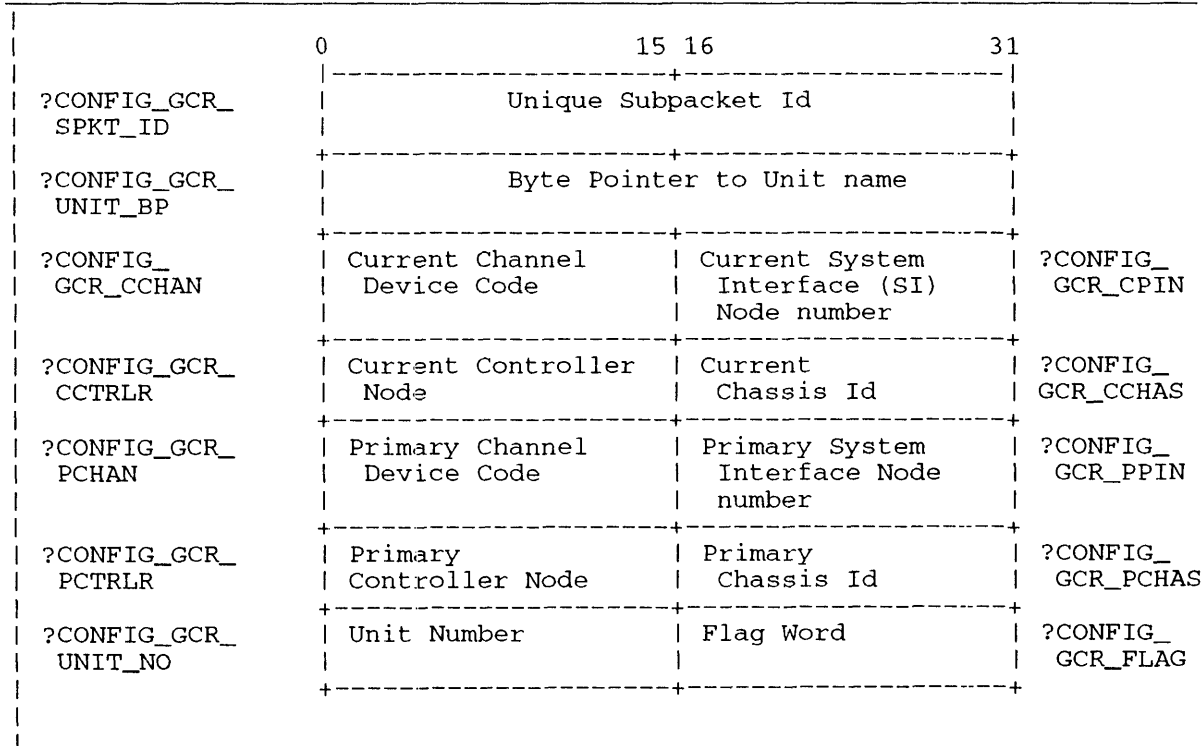


Figure 2-11.2. Structure of the ?CONFIG_GET_CURRENT_ROUTE Function Subpacket

Table 2-8.2. Contents of ?CONFIG_GET_CURRENT_ROUTE Function Subpacket

Offset	Contents
?CONFIG_GCR_SPKT_ID	A unique subpacket identifier (input value).
?CONFIG_GCR_UNIT_BP	A byte pointer to the null terminated unit name string for the MRC device (input value).
?CONFIG_GCR_CCHAN	The channel's device code for the specified unit in the current route (output value).
?CONFIG_GCR_CPIN	The SI node number in the current route to the device (output value).
?CONFIG_GCR_CCTRLR	The controller's node number for the specified unit in the current route (output value).
?CONFIG_GCR_CCHAS	The chassis identifier of the chassis in the current route (output value).
?CONFIG_GCR_PCHAN	The channel's device code for the specified unit in the primary route (output value).
?CONFIG_GCR_PPIN	The SI node number in the primary route to the device (output value).
?CONFIG_GCR_PCTRLR	The controller's node number for the specified unit in the primary route (output value).
?CONFIG_GCR_PCHAS	The primary route chassis identifier (output value).
?CONFIG_GCR_UNIT_NO	The unit number for the specified unit (output value).
?CONFIG_GCR_FLAG	An output flag containing the following bit definitions: ?CONFIG_GCR_RESET_BIT = 0 if no reset operation is pending. ?CONFIG_GCR_RESET_BIT = 1 if there is a reset operation pending. For a disk device, a reset to the new route for the device occurs once the disk is initialized. For tape devices, a reset to the new route occurs when a tape file is next opened on the device.

?CONFIG Continued

What ?CONFIG_RESET_MRC_CHAN Does

The ?CONFIG_RESET_MRC_CHAN function accepts as input the device code of the primary channel whose devices require resetting. After hardware repair, resetting the route on the failed channel results in diverted devices on that channel being rerouted to their primary routes.

?CONFIG_RESET_MRC_CHAN immediately tries to reroute all active diverted devices initialized or opened by a ?GOPEN system call. The reset request does not affect current devices using primary routes.

For inactive MRC disk and tape devices, ?CONFIG_RESET_MRC_CHAN sets an internal flag so that the device route is reset when the device is next accessed. The operating system detects the flag and reroutes the device as soon as a newly initialized disk is accessed, or a file is opened on the tape device.

The ?CONFIG_RESET_MRC_CHANNEL subpacket structure is shown in Figure 2-11.3, and its contents are described in Table 2-8.3.

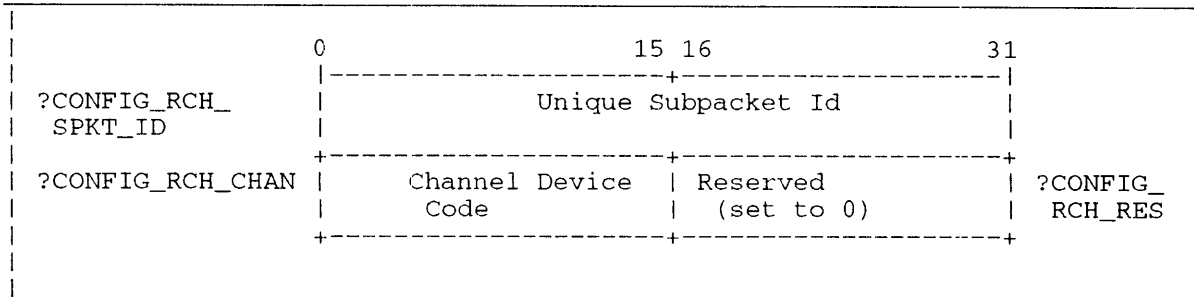


Figure 2-11.3. Structure of the ?CONFIG_RESET_MRC_CHANNEL Function Subpacket

Table 2-8.3 ?CONFIG_RESET_MRD_CHANNEL Function Subpacket Contents

Offset	Contents
?CONFIG_RCH_SPKT_ID	A unique subpacket identifier (an input value).
?CONFIG_RCH_CHAN	The to be reset primary channel's device code (an input value).
?CONFIG_RCH_RES	Reserved (set to 0).

What ?CONFIG_RESET_MRC_CTRLR does

The ?CONFIG_RESET_MRC_CTRLR function accepts as input the channel's device code, and the primary controller's node number for the controller whose devices require resetting. This function operates the same as the ?CONFIG_RESET_MRC_CHAN function. The channel's device code is required to uniquely identify a controller's node when a configuration contains multiple chassis.

For inactive MRC disk and tape devices, ?CONFIG_RESET_MRC_CTRLR sets an internal flag so that the device route is reset when the device is next accessed. The operating system detects the flag and reroutes the device as soon as a newly initialized disk is accessed, or a file is opened on the tape device.

The ?CONFIG_RESET_MRC_CONTROLLER subpacket structure is shown in Figure 2-11.4, and its contents are described in Table 2-8.4.

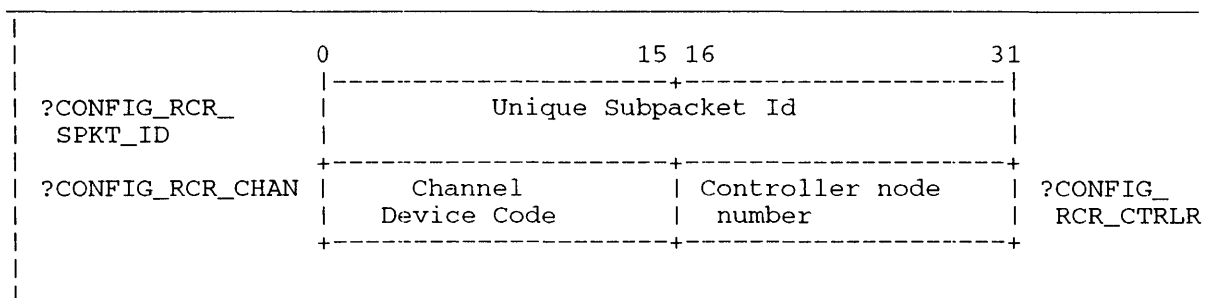


Figure 2-11.4. Structure of the ?CONFIG_RESET_MRC_CTRLR Function Subpacket

Table 2-8.4 ?CONFIG_RESET_MRC_CONTROLLER Function Subpacket Contents

Offset	Contents
?CONFIG_RCR_SPKT_ID	A unique subpacket identifier (an input value).
?CONFIG_RCR_CHAN	Is the channel's device code in the connection between the host and the controller whose devices require resetting (an input value).
?CONFIG_RCR_CTRLR	The primary controller's node number of the controller whose devices require resetting (an input value).

Notes

You can issue ?CONFIG_RESET_MRC_CHAN and ?CONFIG_RESET_MRC_CTRLR calls against a tape device while it is in use, and the call will not affect the active I/O on the tape device. To verify the reset occurred after the subsequent access to the disk or tape device, issue a ?CONFIG with the ?CONFIG_GET_CURRENT_ROUTE subpacket function.

?CONINFO

Request for addressing information
on a terminal or console.

AOS/VS II

?CONINFO [packet address]

error return

normal return

Input

AC0 Reserved (set to zero)
AC1 Reserved (set to zero)
AC2 Address of the ?CONINFO main
packet, unless specified as an
argument to ?CONINFO

Output:

AC0 Unchanged or error code
AC1 Unchanged
AC2 Unchanged

Error codes returned in AC0

ERICN Illegal channel number
ERIFD Invalid function for this device
ERIFT Illegal file type
ERPKT Illegal packet ID
ERPRE Illegal system call parameter
ERPRV Caller not privileged for this action
ERVBP Invalid byte pointer passed as a system
call argument
ERIRB Insufficient room in buffer
ERVWP Invalid word pointer passed as a system
call argument

Why Use It?

Use ?CONINFO to get console addressing information. Due to the layering of some network protocols, ?CONINFO may only resolve information from the caller's host or the last host that routed the connection. Any console generated by VSGEN in the :PER directory can be verified with ?CONINFO. The console does not have to be active.

Who Can Use It?

The ?CONINFO system call is a 32-bit system call, and is for use only in AOS/VS II. The caller must be PID 2, or have the System Manager privilege turned on, or be the process that owns the console.

What It Does

The ?CONINFO system main packet contains input and output parameters. On input, you provide the unique packet id, a pointer to a buffer for return data, the length of the return data buffer, the channel number of the target console (or a byte pointer to the name of the target console), and a flag specifying which target parameter you supplied.

In addition, if you supply a byte pointer, you must include the length of the console name buffer. The return data buffer must be at least ?CON_UBUF_LEN words long, and is defined in PARU_LONG.SR.

On output, the ?CONINFO main packet lists the amount of data placed in the return buffer. Errors are returned in AC0. The return data buffer contains the requested information.

Figure 2-11.5 contains the ?CONINFO main packet structure, and Table 2-8.5 lists the contents of the ?CONINFO main packet.

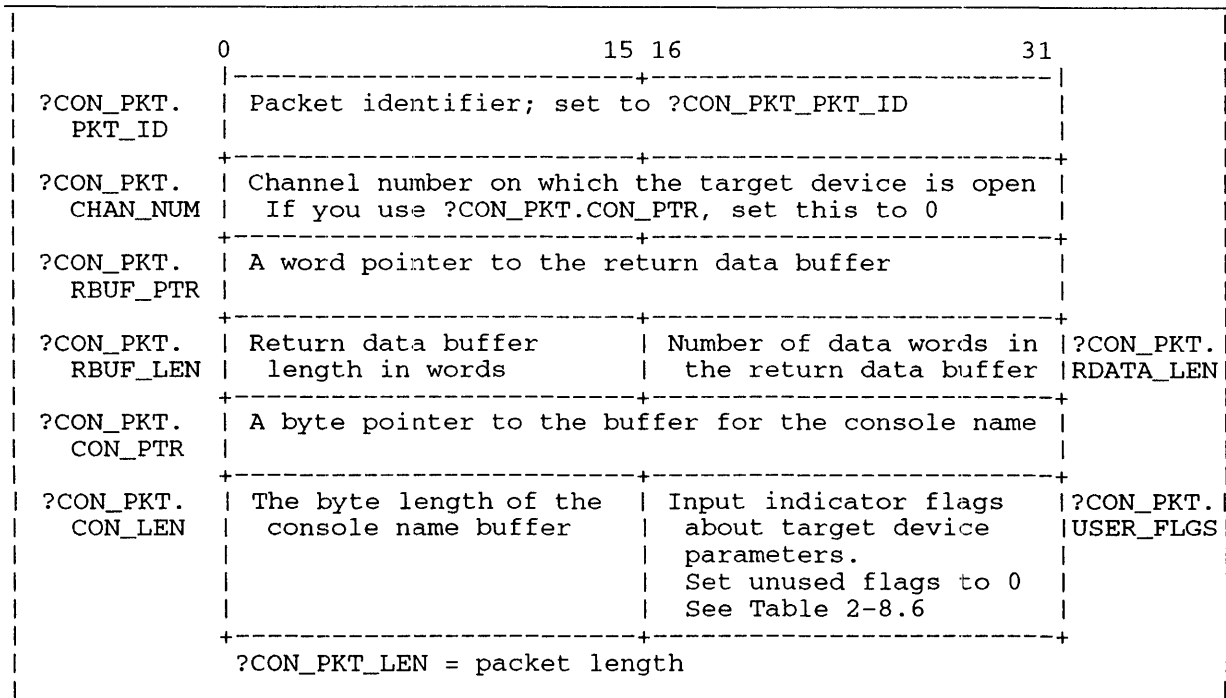


Figure 2-11.5. Structure of ?CONINFO Main Packet

Table 2-8.5. Contents of the ?CONINFO Packet

Offset	Contents
?CON_PKT.PKT_ID	?CON_PKT_PKTID is a unique packet identifier.
?CON_PKT.CHAN_NUM	The channel number on which the target device is open. When using this parameter, ?CON_PKT.CON_PTR and ?CON_PKT.CON_LEN must be zero or an ERPRE error will result.
?CON_PKT.RBUF_PTR	A word pointer to a buffer for return data.
?CON_PKT.RBUF_LEN	The word length of the return data buffer.
?CON_PKT.RDATA_LEN	The number of words of data in the return data buffer.
?CON_PKT.CON_PTR	A byte pointer to the name of the target console. When using this parameter, ?CON_PKT.CHAN_NUM must be set to zero or an ERPRE error will result.
?CON_PKT.CON_LEN	The byte length of the buffer holding the console name. When this parameter is supplied ?CON_PKT.CHAN_NUM must be set to zero or an ERPRE error will result.

?CONINFO Continued

Table 2-8.6. Input Values to ?CON_PKT.USER_FLGS Offset

Offset	Contents
?CON_PKT. USER_FLGS	An input flag value indicating that either a byte pointer to the target device name, or the channel number parameter, was supplied. Unused flags must be set to zero or you generate an ERPRE error. The valid flags are: - ?CON_PKT.USER_FLGS.NAME = 1 means the user has supplied a byte pointer to the name of the target device. If this bit is set, the channel number, ?CON_CHAN_NUM, must be set to zero. - ?CON_PKT.USER_FLGS.CHAN = 1 means the user supplied the channel number of the target device. If this bit is set, the byte pointer ?CON_CON_PTR, and the length of the name buffer, ?CON_CON_LEN, must both be set to zero.

Return Packet Types

In each return packet in the return data buffer, a value in the first offset (word 0) denotes the type of active connection, or the type of session and session's console. Connection and session types, and their values, are defined in Table 2-8.7.

Return Packets and Line Numbers

Each return packet also contains the Terminal Services (TS) target console line number. In the packet for "soft" controllers, the line numbers are zero relative. Thus, when VSGEN generates "150." Telnet consoles for a Telnet connection across a LAN, the Telnet line numbers range from 0 to 149.

Line numbers for directly connected IAC consoles are relative to each TS engine. Thus when VSGEN generates 128 Intelligent TermController (ITC) TermServer consoles, each of the four TS engines on the ITC owns 32 lines, and each engine's line numbers range from 0 to 31.

Table 2-8.7. ?CON_RET_TYPES Return Buffer Console Types and Definitions

Offset	Description
Word 0	<p>?CON_TCP_RET_TYPE = 1 indicates that the target console is a TermServer console connected through an Intelligent TermController (ITC), or a local-bus terminal controller (LTC) running Transport Control Protocol/Internet Protocol (TCP/IP) software.</p> <p>?CON_XNS_RET_TYPE = 2 indicates that the target console is a TermServer console connected through an ITC or LTC controller running Xerox Network Services (XNS) software.</p> <p>?CON_CON_RET_TYPE = 6 indicates that the target console is a CON. These consoles may be Intelligent Asynchronous Controller (IAC) consoles, Ring0 Dual Universal Asynchronous Receiver and Transmitter (DUART) consoles, operator's consoles, or consoles using teletype input and teletype output (TTI-TTO).</p> <p>?CON_TNET_RET_TYPE = 11 indicates that the target console is a TCP/IP connection (TCON) established over Ring0 Telnet software.</p> <p>?CON_ITC_MIN_DATA = 14 indicates no network data is available on an ITC/LTC connection, because of some error on the transport engine.</p> <p>?CON_TSC_MIN_DATA = 15 indicates no network data is available for a ring0 (TCON) network connection.</p> <p>?CON_PVC_RET_DATA = 16 indicates that the target console is an ITC or LTC controller, and the connection is a permanent virtual circuit (PVC) defined on the controller.</p>

?CON_TCP_RET_TYPE Return Type = 1

AOS/VS II returns the packet for a TermServer connection established through an ITC or an LTC controller using TCP/IP. The TCP/IP packet is shown in Figure 2-11.6, and the packet contents are in Table 2-8.8.

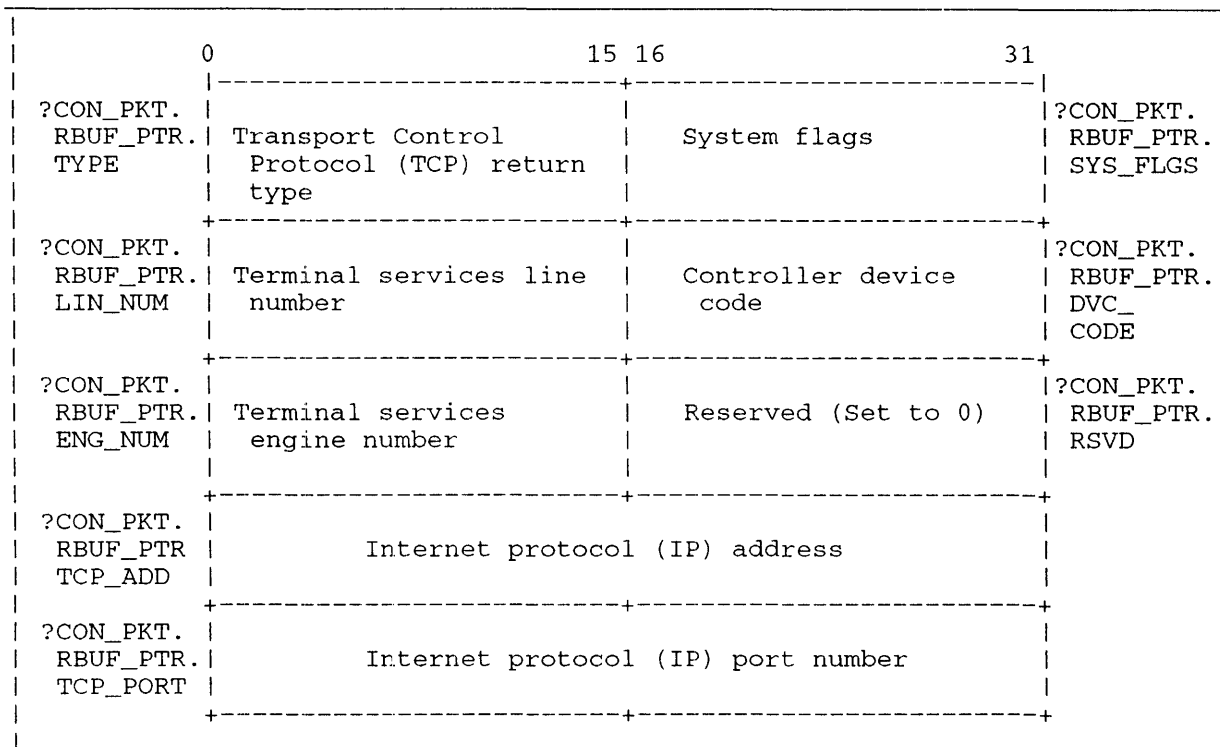


Figure 2-11.6. Structure of ?CON_TCP_RET_TYPE Return Packet

Table 2-8.8. Contents of ?CON_TCP_RET_TYPE Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_TCP_RET_TYPE = 1 identifies the type of packet returned (see Table 2-8.7 for a list of types).
?CON_PKT.RBUF_PTR.SYS_FLGS	Valid only for ?CON_CON_RET_TYPE = 6. Otherwise zero is returned.
?CON_PKT.RBUF_PTR.LIN_NUM	The console terminal services line number in decimal.
?CON_PKT.RBUF_PTR.DVC_CODE	The device code of the controller in octal.
?CON_PKT.RBUF_PTR.ENG_NUM	The terminal services controller engine number that owns the line (ranges from 0 to n).
?CON_PKT.RBUF_PTR._RSVD	Reserved. (Set to 0)
?CON_PKT.RBUF_PTR.TCP_ADD	The caller's IP address in decimal. The address is in four contiguous bytes. Each byte contains a decimal triplet from 000 to 255, with the most significant byte on the left. For example, 107.212.019.036 is an IP address.
?CON_PKT.RBUF_PTR.TCP_PORT	The caller's IP port number in decimal.

?CONINFO Continued

?CON_XNS_RET_TYPE Return Type = 2

AOS/VS II returns this packet for a TermServer connection established through an ITC or an LTC controller using Xerox Network Services (XNS).

The packet structure is shown in Figure 2-11.7. Table 2-8.9 describes each offset.

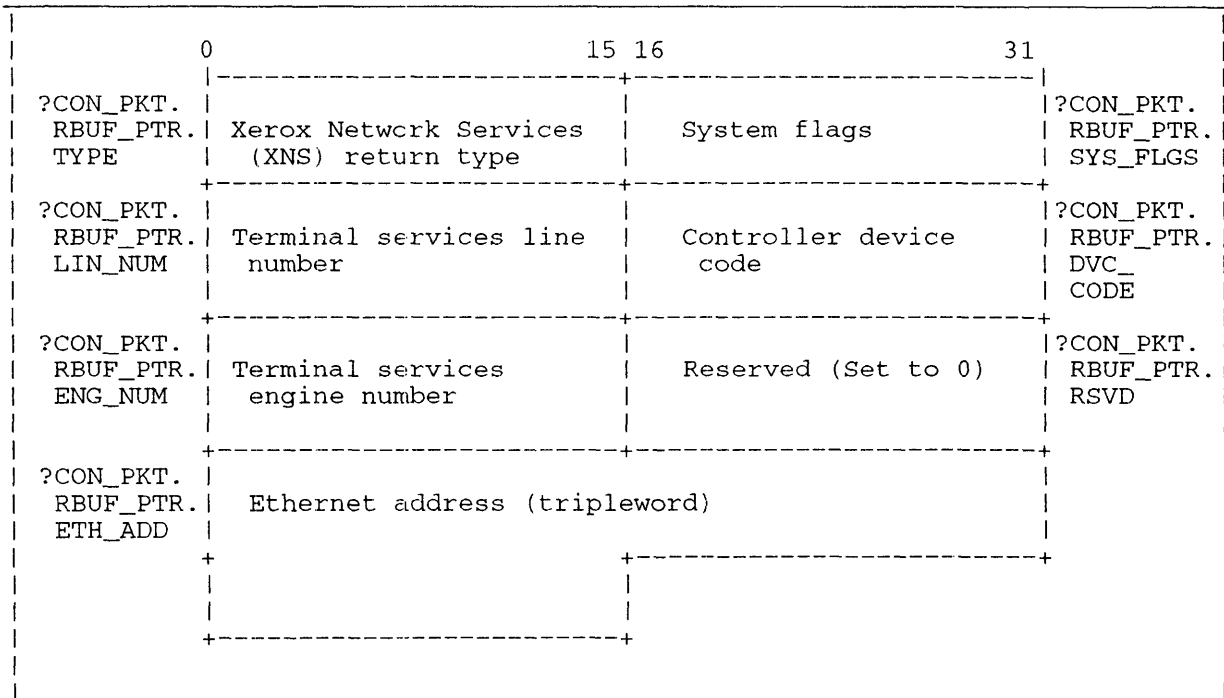


Figure 2-11.7. Structure of ?CON_XNS_RET_TYPE Return Packet

Table 2-8.9. Contents of ?CON_XNS_RET_TYPE Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_XNS_RET_TYPE = 2 identifies the type of packet returned (see Table 2-8.7 for a list of types).
?CON_PKT.RBUF_PTR.SYS_FLGS	Valid only for ?CON_CON_RET_TYPE = 6. Otherwise zero is returned.
?CON_PKT.RBUF_PTR.LIN_NUM	The console terminal services line number in decimal.
?CON_PKT.RBUF_PTR.DVC_CODE	The device code of the TermServer controller in octal.
?CON_PKT.RBUF_PTR.ENG_NUM	The terminal services controller engine number that owns the line (ranges from 0 to n).
?CON_PKT.RBUF_PTR.RSVD	Reserved. (Set to 0)
?CON_PKT.RBUF_PTR.ETH_ADD (tripleword)	The Ethernet address of the TermServer box--12 hexadecimal digits.

?CON_CON_RET_TYPE Return Type = 6

AOS/VS II returns this packet for consoles directly connected to IACs, modern lines, Ring0 DUARTS, Opcons, and consoles using TTI-TTO. The packet structure is shown in Figure 2-11.8. Table 2-8.10 describes each offset.

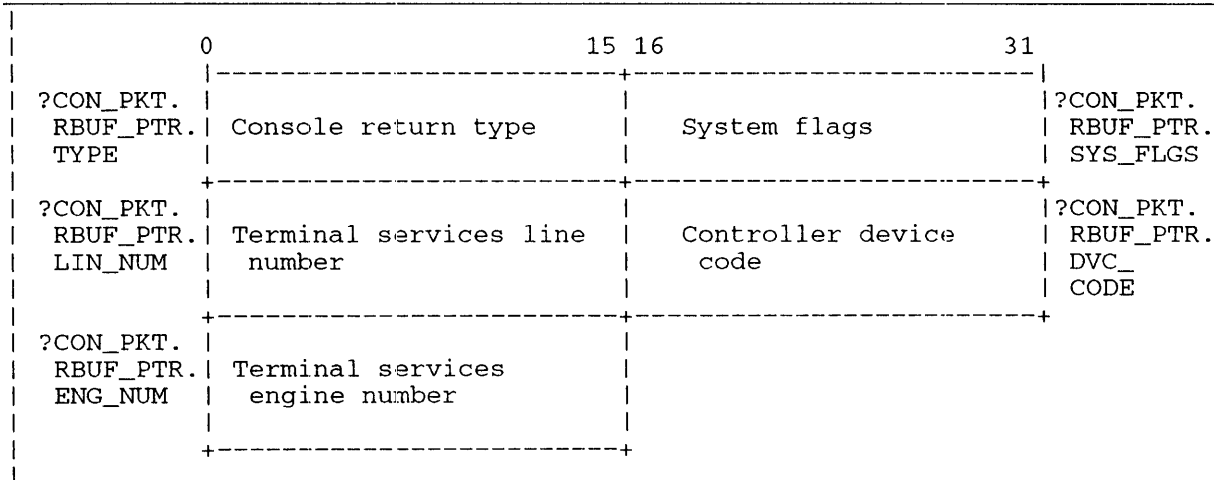


Figure 2-11.8. Structure of ?CON_CON_RET_TYPE Return Packet

?CONINFO Continued

Table 2-8.10. Contents of ?CON_CON_RET_TYPE Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_CON_RET_TYPE = 6 identifies the type of packet returned (see Table 2-8.7 for a list of types).
?CON_PKT.RBUF_PTR.SYS_FLGS	Returns one of the following values: ?CON_PKT.RBUF_PTR.SYS_FLGS.OPCON = 1 if the target console is the system operator's console. ?CON_PKT.RBUF_PTR.SYS_FLGS.MODEM = 1 if the target console's modem characteristic is switched on.
?CON_PKT.RBUF_PTR.LIN_NUM	The terminal services console line number in decimal.
?CON_PKT.RBUF_PTR.DVC_CODE	The controller device codes, for example DUART = 34 (octal) TTI-TTO = 11 (octal) IAC = 41 (octal)
?CON_PKT.RBUF_PTR.ENG_NUM	The terminal services controller engine number that owns the line (ranges from 0 to n).

?CON_TNET_RET_TYPE Return Type = 11

The Ring0 Telnet connection (TCON) return packet is shown in Figure 2-11.9, and the packet contents are in Table 2-8.11.

	0	15 16	31
?CON_PKT.RBUF_PTR.TYPE	Telnet return type	System flags	?CON_PKT.RBUF_PTR.SYS_FLGS
?CON_PKT.RBUF_PTR.LIN_NUM	Terminal services line number.	Reserved (set to 0)	?CON_PKT.RBUF_PTR.TNET_RSVD
?CON_PKT.RBUF_PTR.IP_ADD	Internet protocol (IP) address		
?CON_PKT.RBUF_PTR.IP_PORT	Internet protocol (IP) port number		

Figure 2-11.9. Structure of ?CON_TNET_RET_TYPE Return Packet

Table 2-8.11. Contents of ?CON_TNET_RET_TYPE Return Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_TNET_RET_TYPE = 11 identifies the type of packet returned (see Table 2-8.7 for a list of types.)
?CON_PKT.RBUF_PTR.SYS_FLGS	Valid only for ?CON_CON_RET_TYPE = 6. Otherwise zero is returned.
?CON_PKT.RBUF_PTR.LIN_NUM	The console terminal services line number in decimal.
?CON_PKT.RBUF_PTR.TNET_RSVD	Reserved. (Set to 0)
?CON_PKT.RBUF_PTR.IP_ADD	The caller's IP address in decimal. (see page NO TAG for a description).
?CON_PKT.RBUF_PTR.IP_PORT	The caller's IP port number in decimal.

?CON_ITC_MIN_DATA Return Type = 14

AOS/VS II returns the ?CON_ITC_MIN_DATA packet for a TermServer console established through an ITC or an LTC controller using XNS or TCP when no network connection data is available. The condition can occur when there is no connection on the TermServer line, or when there is a connection, but the transport created the connection without any ?CONINFO data for the line. Revision incompatibility between TS software and your TermServer network software can also produce this condition.

The ?CON_ITC_MIN_DATA packet structure is shown in Figure 2-11.10. Table 2-8.12 describes each offset.

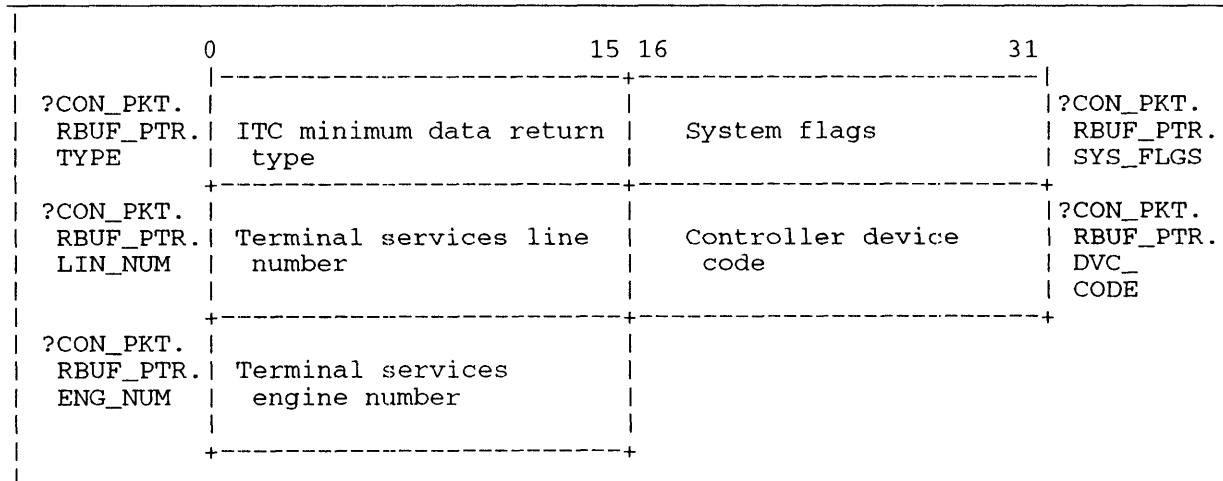


Figure 2-11.10. Structure of ?CON_ITC_MIN_DATA Return Packet

?CONINFO Continued

Table 2-8.12. Contents of ?CON_ITC_MIN_DATA Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_ITC_MIN_DATA = 14 identifies the type of packet returned (see Table 2-8.7 for a list of types).
?CON_PKT.RBUF_PTR.SYS_FLGS	Valid only for ?CON_CON_RET_TYPE = 6. Otherwise zero is returned.
?CON_PKT.RBUF_PTR.LIN_NUM	The console terminal services line number in decimal.
?CON_PKT.RBUF_PTR.DVC_CODE	The device code of the TermServer controller in octal.
?CON_PKT.RBUF_PTR.ENG_NUM	The terminal services controller engine number that owns the line (ranges from 0 to n).

?CON_TSC_MIN_DATA Return Type = 15

AOS/VSI returns the ?CON_TSC_MIN_DATA packet for a Ring0 “soft” controller (TCON) when there is no connection on the console and no network information available.

The ?CON_TSC_MIN_DATA packet structure is shown in Figure 2-11.11, and Table 2-8.13 describes each offset.

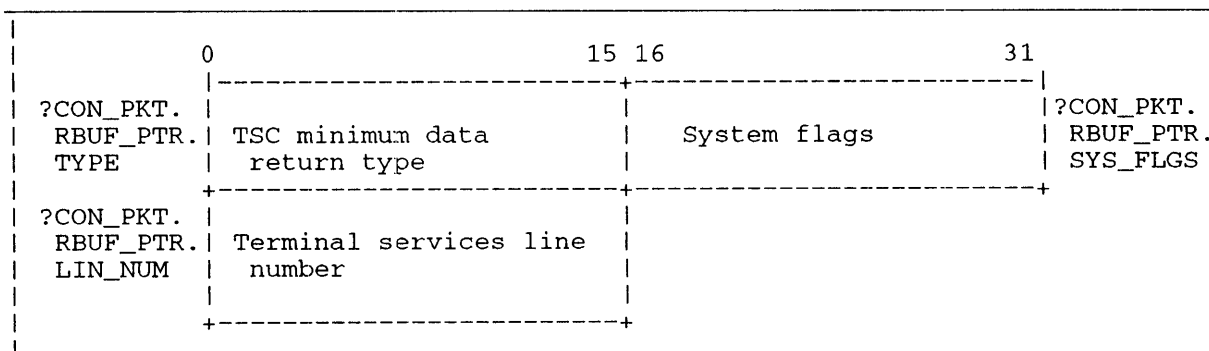


Figure 2-11.11. Structure of ?CON_TSC_MIN_DATA Return Packet

Table 2-8.13. Contents of ?CON_TSC_MIN_DATA Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_TSC_MIN_DATA = 15 identifies the type of packet returned (see Table 2-8.7 for a list of types).
?CON_PKT.RBUF_PTR.SYS_FLGS	Valid only for ?CON_CON_RET_TYPE = 6. Otherwise zero is returned.
?CON_PKT.RBUF_PTR.LIN_NUM	The console terminal services line number in decimal.

?CON_PVC_RET_TYPE Return Type = 16

AOS/VS II returns the ?CON_PVC_RET_TYPE packet for a TermServer connection established through an ITC or an LTC controller using PVCs over either the XNS or TCP protocols. The packet contains the controller device code, the number of the TS engine owning the line, and the line number on the engine that owns the console. The packet also contains a PVC subtype field describing the format of the data defining the PVC connection address.

The packet's first seven words are common to all PVC packets. The ?CON_PVC_RET_TYPE packet structure is shown in Figure 2-11.12, and Table 2-8.14 describes each offset.

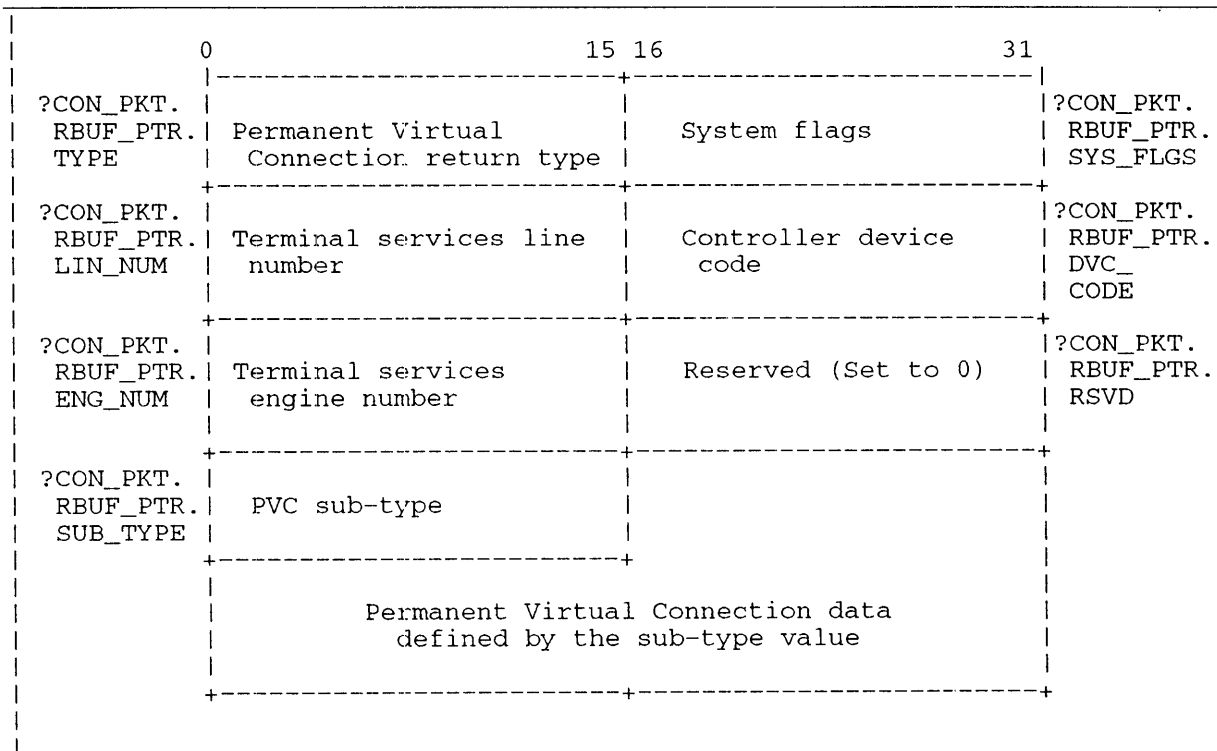


Figure 2-11.12. Structure of ?CON_PVC_RET_TYPE Return Packet

Table 2-8.14. Contents of ?CON_PVC_RET_TYPE Packet

Offset	Contents
?CON_PKT.RBUF_PTR.TYPE	?CON_PVC_RET_TYPE = 16 identifies the type of packet returned (see Table 2-8.7 for a list of types).
?CON_PKT.RBUF_PTR.SYS_FLGS	Valid only for ?CON_CON_RET_TYPE = 6. Otherwise zero is returned.
?CON_PKT.RBUF_PTR.LIN_NUM	The console terminal services line number in decimal.
?CON_PKT.RBUF_PTR.DVC_CODE	The device code of the controller.
?CON_PKT.RBUF_PTR.ENG_NUM	The terminal services controller engine number that owns the line (ranges from 0 to n).
?CON_PKT.RBUF_PTR.RSVD	Reserved. (Set to 0)
?CON_PKT.RBUF_PTR.SUB_TYPE	A value in the subpacket defining one of seven different types of data returned from the ITC describing the PVC connection address. The values are: ?CON_PVC_NAME = 0 ?CON_PVC_NAME_PORT = 1 ?CON_PVC_IP = 2 ?CON_PVC_IP_PORT = 3 ?CON_PVC_ETH = 4 ?CON_PVC_PORT = 5 ?CON_PVC_NET = 6

?CONINFO Continued

?CON_PVC_RET_TYPE Subpackets

The ?CON_PVC_RET_TYPE packet returns a PVC subtype and its corresponding subpacket. The first six words of the subpacket are identical to ?CON_PVC_RET_TYPE. The subtypes begin at Word 7 in each subpacket. In the following sections, the PVC-specific content of the subpackets is enumerated in PVC subtype order.

The subtypes and their values, are defined in Table 2-8.15.

Table 2-8.15. ?CON_PVC_RET_TYPE Subpacket Types and Definitions

Offset	Definitions
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_NAME = 0 is the host name upon which the TermServer console connection is established through an ITC or an LTC controller using PVCs over either an XNS or TCP protocol. The connection's host name is returned. ?CON_PVC_NAME_PORT = 1 is a TermServer connection established through an ITC or an LTC controller using PVCs over an XNS or TCP protocol. The connection's host name and port are returned. ?CON_PVC_IP = 2 is a TermServer connection established through an ITC or an LTC controller using PVCs over TCP protocol. The console's IP address is returned. ?CON_PVC_IP_PORT = 3 is a TermServer connection established through an ITC or an LTC controller using PVCs over TCP protocol. The console's IP address and port are returned. ?CON_PVC_ETH = 4 is a TermServer connection established through an ITC or an LTC controller using PVCs over XNS protocol. The console's Ethernet address is returned. ?CON_PVC_PORT = 5 is a TermServer connection established through an ITC or an LTC controller using PVCs over XNS protocol. The console's port number is returned. ?CON_PVC_NET = 6 is a TermServer connection established through an ITC or an LTC controller using PVCs over XNS protocol. The console's network address, port number, and Ethernet address is returned.

?CON_PVC_NAME Subtype = 0 — The packet contains the first six words as described in Figure 2–11.12 on page 2–58.16. The PVC_specific data in the ?CON_PVC_NAME packet structure are shown in Figure 2–11.13, and Table 2–8.16 describes each offset.

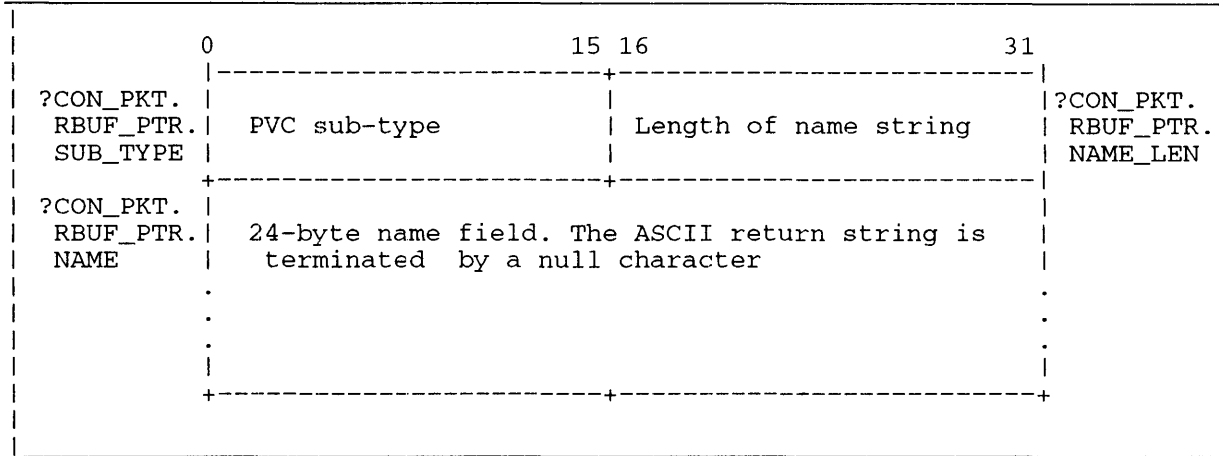


Figure 2–11.13. Structure of ?CON_PVC_NAME Return Packet

Table 2–8.16. Contents of ?CON_PVC_NAME Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_NAME = 0 identifies the type of PVC data (see Table 2-8.15 for a list of types.)
?CON_PKT.RBUF_PTR.NAME_LEN	The length of the ASCII name string in ?CON_PKT.RBUF_PTR.NAME in bytes.
?CON_PKT.RBUF_PTR.NAME	The name bound to an XNS or TCP physical address for the connection. The name is a null-terminated ASCII string with a maximum length of 24 bytes.

?CONINFO Continued

?CON_PVC_NAME_PORT Subtype = 1 — The packet contains the first six words as described in Figure 2–11.12 on page 2–58.16. The PVC-specific data, shown in the ?CON_PVC_NAME_PORT packet structure in Figure 2–11.14. Each offset is described in Table 2–8.17.

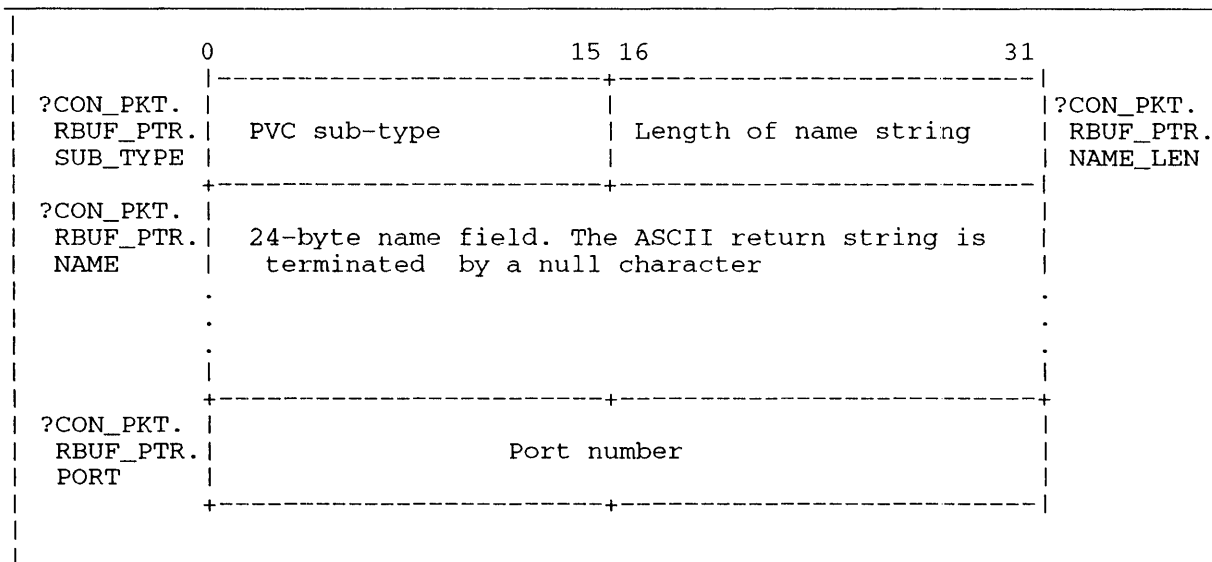


Figure 2–11.14 Structure of ?CON_PVC_NAME_PORT return packet

Table 2–8.17. Contents of ?CON_PVC_NAME_PORT Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_NAME_PORT = 1 identifies the type of PVC data (see Table 2–8.15 for a list of types).
?CON_PKT.RBUF_PTR.NAME_LEN	The length of the ASCII name string in ?CON_PKT.RBUF_PTR.NAME in bytes.
?CON_PKT.RBUF_PTR.NAME	The name bound to an XNS or TCP physical address for the connection. The name is a null terminated ASCII string with a maximum length of 24 bytes.
?CON_PKT.RBUF_PTR.PORT	The port number on the host supporting the TCP physical connection.

?CON_PVC_IP Subtype = 2 — The packet contains the first six words as described in Figure 2-11.12 on page 2-58.16. The PVC-specific data in the ?CON_PVC_IP packet structure are shown in Figure 2-11.15. Each offset is described in Table 2-8.18.

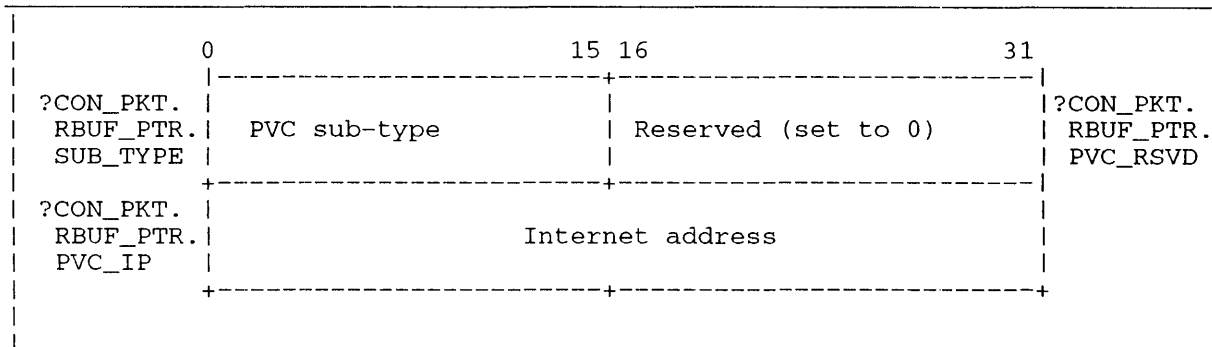


Figure 2-11.15. Structure of ?CON_PVC_IP return packet

Table 2-8.18. Contents of ?CON_PVC_IP Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_IP = 2 identifies the type of PVC data (see Table 2-8.15 for a list of types).
?CON_PKT.RBUF_PTR.PVC_RSVD	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.PVC_IP	The Internet address of the PVC connection. Returned to the packet in four contiguous decimal bytes (See Table 2-8.8 for an explanation).

?CONINFO Continued

?CON_PVC_IP_PORT Subtype = 3 — The packet contains the first six words as described in Figure 2–11.12 on page 2–58.16. The PVC_specific data in the ?CON_PVC_IP_PORT packet structure are shown in Figure 2–11.16. Each offset is described in Table 2–8.19.

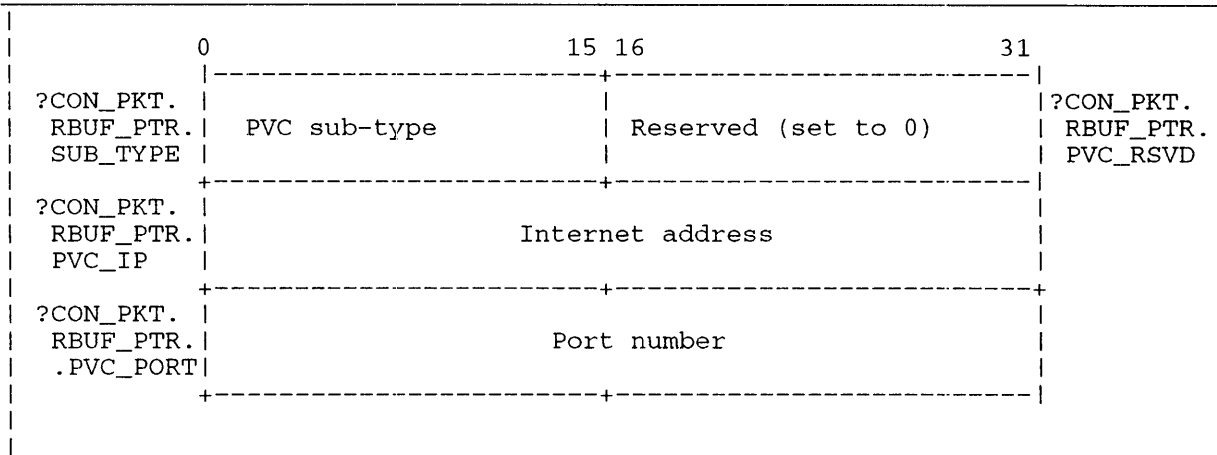


Figure 2–11.16 Structure of ?CON_PVC_IP_PORT return packet

Table 2–8.19. Contents of ?CON_PVC_IP_PORT Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_IP_PORT = 3 identifies the type of PVC data (see Table 2–8.15 for a list of types).
?CON_PKT.RBUF_PTR.PVC_RSVD	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.PVC_IP	The Internet address of the PVC connection. Returned to the packet in four contiguous hexadecimal bytes (See Table 2–8.8 for an explanation).
?CON_PKT.RBUF_PTR.PVC_PORT	The Internet port number of the PVC connection.

?CON_PVC_ETH Subtype = 4 — The packet contains the first six words as described in Figure 2-11.12 on page 2-58.16. The PVC-specific data in the ?CON_PVC_ETH packet structure are shown in Figure 2-11.17. Each offset is described in Table 2-8.20.

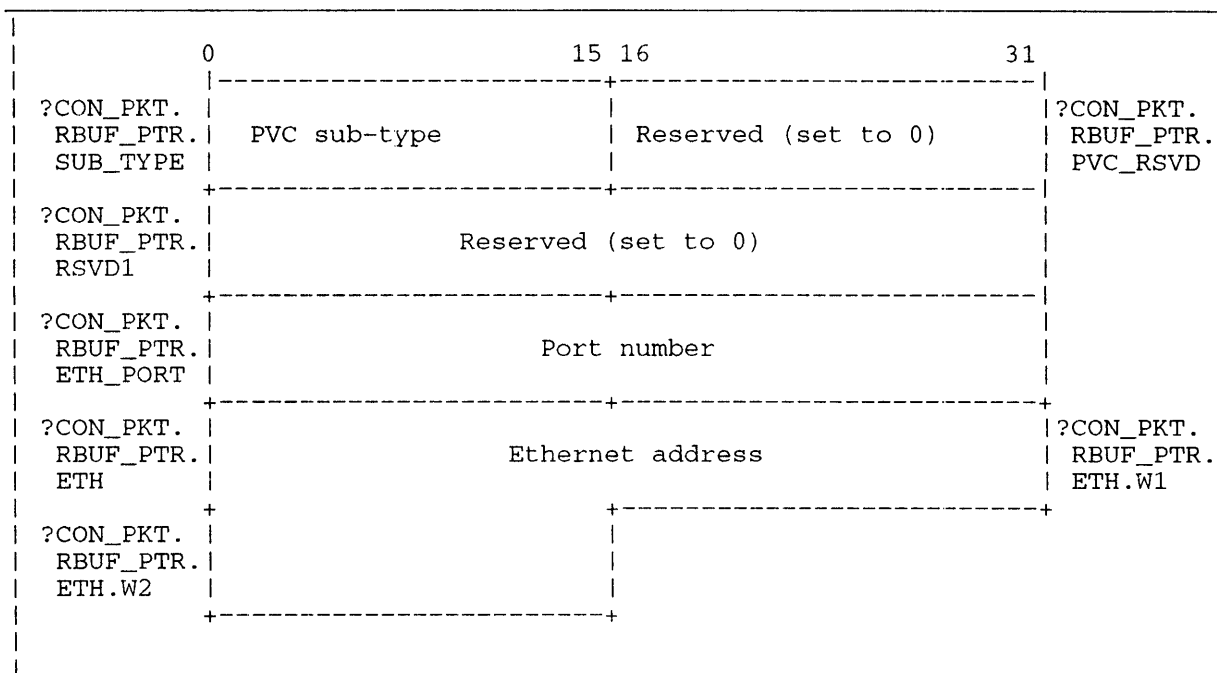


Figure 2-11.17 Structure of ?CON_PVC_ETH return packet

Table 2-8.20. Contents of ?CON_PVC_ETH Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_ETH = 4 identifies the type of PVC data (see Table 2-8.15 for a list of types).
?CON_PKT.RBUF_PTR.PVC_RSVD	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.PVC_RSVD1	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.ETH_PORT	The Ethernet port number of the PVC connection.
?CON_PKT.RBUF_PTR.ETH	The Ethernet address of the PVC connection, returned to the packet in 12 contiguous hexadecimal digits.

?CONINFO Continued

?CON_PVC_PORT Subtype = 5 — The packet contains the first six words as described in Figure 2–11.12 on page 2–58.16. The PVC-specific data in the ?CON_PVC_PORT packet structure are shown in Figure 2–11.18. Each offset is described in Table 2–8.21.

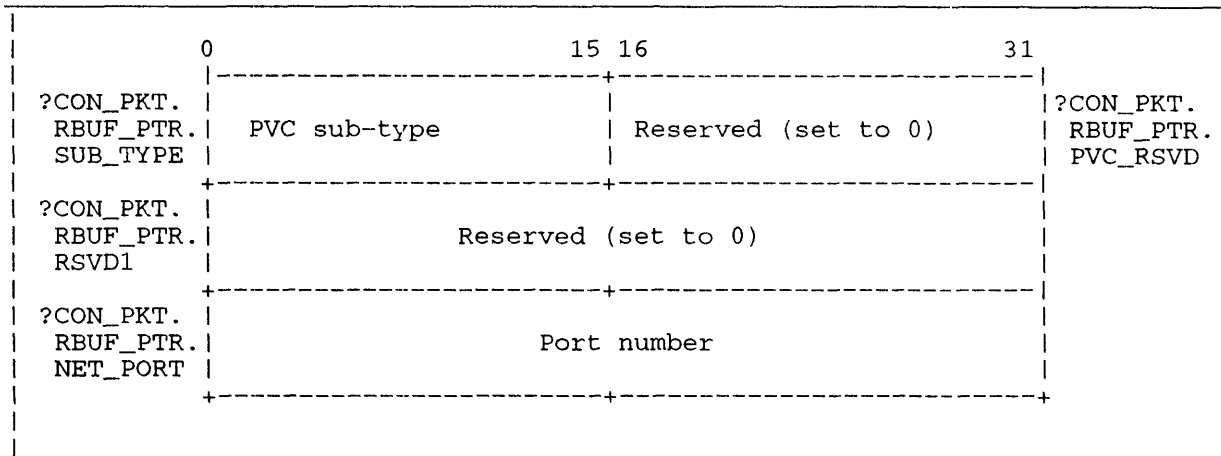


Figure 2–11.18 Structure of ?CON_PVC_PORT return packet

Table 2–8.21. Contents of ?CON_PVC_PORT Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_ETH = 4 identifies the type of PVC data. (See Table 2–8.15 for a list of types.)
?CON_PKT.RBUF_PTR.PVC_RSVD	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.PVC_RSVD1	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.NET_PORT	The Ethernet port number of the PVC connection.

?CON_PVC_NET Subtype = 6 — The packet contains the first six words as described in Figure 2-11.12 on page 2-58.16. The PVC-specific data in the ?CON_PVC_NET packet structure are shown in Figure 2-11.19. Each offset is described in Table 2-8.22.

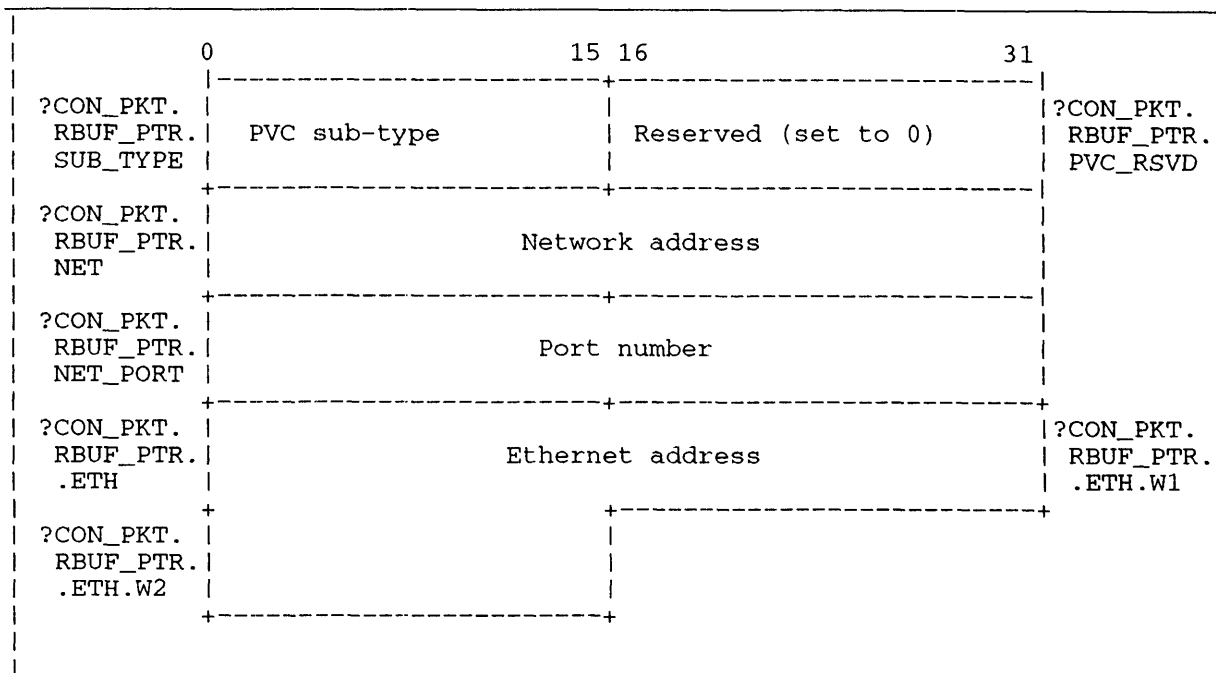


Figure 2-11.19 Structure of ?CON_PVC_NET return packet

Table 2-8.22. Contents of ?CON_PVC_NET Subpacket

Offset	Contents
?CON_PKT.RBUF_PTR.SUB_TYPE	?CON_PVC_NET = 6 identifies the type of PVC data (see Table 2-8.15 for a list of types).
?CON_PKT.RBUF_PTR.PVC_RSVD	Reserved. Set to 0.
?CON_PKT.RBUF_PTR.NET	The Network address of the PVC connection. Returned to the packet in 8 contiguous hexadecimal digits.
?CON_PKT.RBUF_PTR.NET_PORT	The Ethernet port number of the PVC connection.
?CON_PKT.RBUF_PTR.NET.ETH	The Ethernet address of the PVC connection, returned to the packet in 12 contiguous hexadecimal digits.

?CPMAX Sets maximum size for a control point directory (CPD).

?CPMAX [*packet address*]

error return

normal return

Input

Output

AC0	One of the following: <ul style="list-style-type: none">• Byte pointer to the CPD's pathname• 0 if a packet address is supplied	AC0	Unchanged
AC1	One of the following: <ul style="list-style-type: none">• CPD's new maximum space (MS) value if not supplying a packet• Unused if supplying a packet	AC1	Unchanged
AC2	One of the following: <ul style="list-style-type: none">• Reserved (Set to 0 if not supplying a packet.)• Address of the ?CPMAX packet, unless you specify the address as an argument to ?CPMAX	AC2	Unchanged

Error Codes in AC0

ERCPD	CPD maximum size exceeded
ERIFT	Illegal file type (The target is not a CPD or is the root.)
ERVBP	Invalid byte pointer passed as a system call argument
ERWAD	Write access denied
ER_FS_DIRECTORY_NOT_AVAILABLE	Directory not available because the file system is force released (AOS/VS II only)
ER_FS_TLA_MODIFY_VIOLATION	Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet

Why Use It?

?CPMAX lets you restrict the space that is assigned to a CPD and its subordinate directories to a predefined limit. This is useful for managing and/or conserving your system's disk space.

Who Can Use It?

There are no special process privileges needed to issue this call. The calling process must have Write access to the target CPD.

What It Does

?CPMAX changes the maximum space (MS) value of the target CPD. The target CPD can be specified in one of two ways: either by a byte pointer to the CPD's pathname in AC0, or by using

Number of Primary Elements = 1.

Index Element Size = 1.

- AOS/VS II assigns the following additional file structure parameters to a directory that you create with ?CREATE:

Primary Data Element Size = 1.

Maximum Index Level = Currently defined by the system at 3.

- For AOS/VS II, see the description of the ?XCREATE and ?XFSTAT system calls for the additional file structuring parameters that the ?XCREATE system call sets and the ?XFSTAT call displays.

?CRUDA

Creates a user data area (UDA).

?CRUDA [*packet address*]

error return

normal return

Input

- AC0 One of the following:
- Byte pointer to the pathname of the target
 - 0 if a packet address is supplied
- AC1 Reserved (Set to 0.)
- AC2 One of the following:
- Reserved (Set to 0 if not supplying a packet.)
 - Address of the ?CRUDA packet, unless you specify the address as an argument to ?CRUDA

Output

- AC0 Unchanged
file
- AC1 Unchanged
- AC2 Undefined or address of ?CRUDA packet

Error Codes in AC0

- ERFAD File access denied
- ERIFT Illegal file type
- ERNRD Insufficient room in directory
- ERUAE User data area already exists
- ERVBP Invalid byte pointer passed as a system call argument
- ERVWP Invalid word pointer passed as a system call argument
- ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)
- ER_FS_INVALID_PATHNAME_BYTE_PTR
Invalid byte pointer to pathname
- ER_FS_TLA_MODIFY_VIOLATION
Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet

Why Use It?

You can use UDA data to tailor a file's output specifications, provided the intended output device is a data channel line printer that is controlled by the EXEC utility. If there is no UDA defined for a file, the operating system uses the default EXEC format specifications.

Who Can Use It?

There are no special process privileges needed to issue this call. If you specified the file with a channel number, you must have Write or Owner access to the target file. If, on the other hand, you specified the file with a pathname, you must also have Execute access to the parent directory.

?DELAY

Suspends a 16-bit task for a specified interval (16-bit processes only).

?DELAY

error return

normal return

Input

AC0 and AC1

Delay interval (in milliseconds) in Bits 16 through 31 of each accumulator

AC2 Reserved (Set to 0.)

Output

AC0 and AC1

Modified (The OS uses these accumulators to maintain delays.)

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?DELAY allows a task to wait until a specified amount of time has passed.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?DELAY, which is the 16-bit counterpart of ?WDELAY, suspends the calling task for the number of milliseconds that you specify in AC0 and AC1. You must express the delay interval as a double-precision integer in Bits 16 through 31 of AC0 and AC1.

If the number of milliseconds that you specify is not a multiple of the real-time clock frequency, the operating system rounds it to the next highest frequency multiple. For example, if the clock frequency is 10 hertz (one period = 100 milliseconds) and you specify a delay of 120 milliseconds, the operating system rounds the delay interval to 200 milliseconds. To obtain the real-time clock frequency, issue ?GHRZ.

Notes

- See the descriptions of ?WDELAY and ?GHRZ in this chapter.

?DELETE

Deletes a file entry.

?DELETE [*packet address*]

error return

normal return

Input

- AC0 One of the following:
- Byte pointer to the pathname of the target file
 - 0 if a packet address is supplied

AC1 Reserved (Set to 0.)

- AC2 One of the following:
- Reserved (Set to 0 if not supplying a packet.)
 - Address of the ?DELETE packet, unless you specify the address as an argument to ?DELETE

Output

AC0 Unchanged

AC1 Undefined

AC2 Undefined or address of ?DELETE packet

Error Codes in AC0

- ERDID Directory delete error (You tried to delete a directory that contains one or more subordinate directories.)
- ERDIU Directory in use — cannot delete (You tried to delete a working directory or a directory cited in a search list.)
- ERFDE File does not exist
- ERPRM Cannot delete permanent file (You tried to delete a file or directory with the Permanence attribute.)
- ERVBP Invalid byte pointer passed as a system call argument
- ERWAD Write access denied
- ERCDCR Can't delete the root directory
- ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)
- ER_FS_TLA_MODIFY_VIOLATION
Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet

Why Use It?

By deleting a file or directory, you can reclaim disk space or, in the case of an IPC file, renumber the port.

When you issue ?DELETE against an open file, the operating system deletes the filename immediately, and then waits until all users have closed the file before it actually deletes the file. ?DELETE also deletes any user data area (UDA) associated with the file. If the input pathname does not begin with a prefix character (:, ^, or =), ?DELETE assumes a prefix of =. This means that ?DELETE will not scan the caller's search list to find the target file.

?DIR

Changes the working directory.

?DIR

error return

normal return

Input

- AC0 One of the following:
- Byte pointer to the pathname of the new working directory (that is, the destination directory)
 - 0 to specify the initial working

Output

- AC0 Unchanged
- AC1 Reserved (Set to 0.) Undefined
- AC2 Reserved (Set to 0.) Undefined
- directory

Error Codes in AC0

- ERDAD Directory access denied
- ERVBP Invalid byte pointer passed as a system call argument
- ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)
- ER_FS_INVALID_PATHNAME_BYTE_PTR
Invalid byte pointer to pathname

Why Use It?

A process's working directory is its starting point for file access. You can change your working directory to access files that are not contained in your current working directory. ?DIR also lets you return to your initial directory after you work elsewhere.

Who Can Use It?

There are no special process privileges needed to issue this call. The calling process must have Execute access to a directory to use it as a working directory. The calling process must also have Execute access to the target directory's parent directory.

What It Does

?DIR changes the caller's working directory to the directory that you specify in AC0 or to the caller's initial working directory (if AC0 contains 0).

?DQTSK

Removes from the queue one or more previously queued tasks.

?DQTSK *[task definition packet address]*

error return

normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Address of the task definition packet to remove from the initiation queue, unless you specify the address as an argument to ?DQTSK

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the task definition packet

Error Codes in AC0

ERVWP	Invalid word pointer passed as a system call argument
ERQTS	Error in qtask request (There is no match on the task definition packet.)

Why Use It?

?DQTSK can be useful for error handling, because it lets you remove one or more tasks previously placed on the execution queue with ?IQTSK and ?TASK. ?DQTSK has no effect if the task has already begun to execute.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?DQTSK removes a specific task or tasks (as defined in the task definition packet) from the initiation queue you previously established with ?IQTSK.

?DQTSK takes the same task definition packet that you passed to ?TASK when you initiated the target task or tasks. (See Figure 2–19.) You can specify the packet address as an argument to ?DQTSK, or you can load the address into AC2 before you issue ?DQTSK. You should not alter the task definition packet before you issue ?DQTSK.

?ESFF

Flushes shared file memory pages to disk.

?ESFF

error return

normal return

Input

AC0 Reserved (Set to 0.)
AC1 Channel number of the file
to be flushed
AC2 Reserved (Set to 0.)

Output

AC0 Unchanged
AC1 Unchanged
AC2 Undefined

Error Codes in AC0

ERICN Illegal channel
ERSIM Simultaneous I/O
ERIFT Illegal file type
ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)
ER_FS_TLA_MODIFY_VIOLATION
Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet

Why Use It?

The ?ESFF call ensures that if the system fails, you do not lose any changes to shared files up to the time you issued ?ESFF. When you issue ?ESFF, the operating system saves all your changes in the disk file. This means that in an emergency situation, you will lose only those changes you made to a shared file after you issued ?ESFF (provided the ?ESFF completes without error).

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?ESFF flushes modified shared pages to disk. Even when an I/O error occurs on one or more of the modified pages, the operating system tries to flush the remaining pages.

When an ?ESFF completes successfully, all pages that were modified at the start of the ?ESFF are on disk. However, if another process modifies a page after the ?ESFF begins, that page may or may not be flushed to disk when the ?ESFF completes.

Note, however, that ?ESFF does not ensure that you'll be able to access the flushed pages; use ?UPDATE to write out updated file descriptor information for the shared file.

Notes

- See the description of ?UPDATE in this chapter.

?EXEC

Requests a service from EXEC.

AOS/VS

?EXEC [*packet address*]

error return

normal return

Input

AC0	Reserved (Set to 0.)
AC1	Host ID for Resource Management Agent (RMA) deflection, otherwise 0.
AC2	Address of the ?EXEC packet, unless you specify the address as an argument to ?EXEC

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?EXEC packet

Error Codes in AC0

EREGN	End of get next sequence
ERRBO	(Request) refused by operator
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument
ERWMT	Volume not mounted
ERXNA	EXEC not available
ERXSP	Invalid EXEC string parameter
ERXUF	EXEC request function unknown (You passed an unknown value in offset ?XRFNC.)

Why Use It?

You can use ?EXEC to request a service, such as mounting a labeled tape, from the EXEC utility.

Who Can Use It?

One special privilege needed to issue this call is the batch usage privilege (value ?PBCHPRV) for submitting batch jobs (see the section "Queuing a File Entry"). The only other special privileges needed are Write access to a queue to submit it, and Read access to a queue to issue a QDISPLAY command against it. There are no restrictions concerning file access.

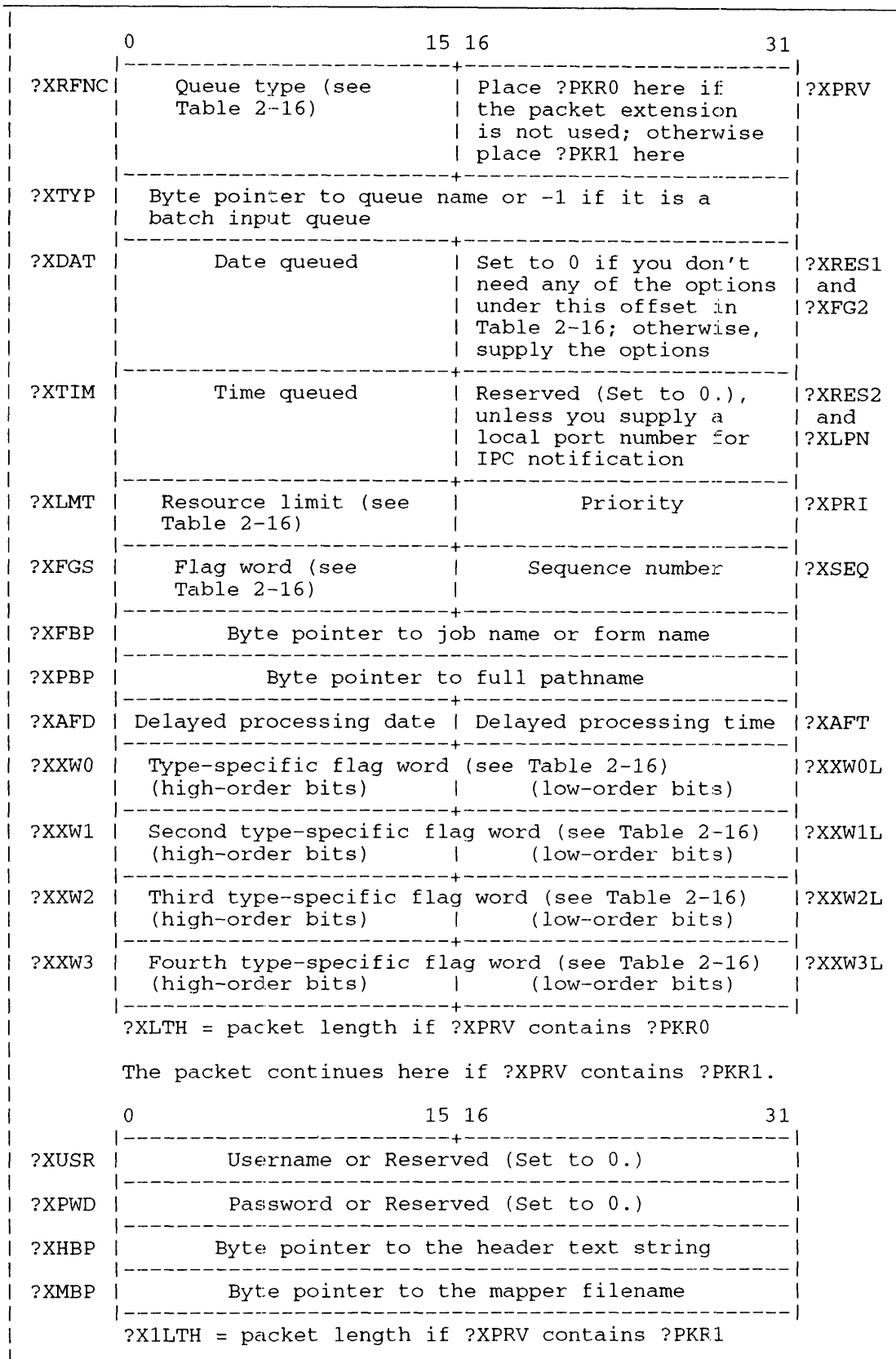


Figure 2-28. Structure of ?EXEC Packet for Queue Requests

?EXEC Continued

Table 2-16. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XRFNC	<p>Queue type: Place entry into one of the following:</p> <ul style="list-style-type: none"> ?XFBAT--Batch queue. ?XFFTA--FTA queue. ?XFHAM--HAMLET queue. ?XFLPT--Print queue. ?XFPLT--Plot queue. ?XFSNA--SNA/RJE queue. ?XFSUB--Any type of queue (except ?XFUSR). ?XFOTH--Submit job for another user. <p>Setting Bit 0 of ?XRFNC indicates that RMA should deflect this request to the remote host specified in AC1.</p> <p>A ?XFBAT request is, other than the value in offset ?XRFNC, identical to a generic ?XFSUB queue submission call. Function code ?XFBAT is provided to allow for further specialization and expansion of batch queue submission.</p>
?XPRV	<p>Packet revision number. (Set to ?PKF1 if offset ?XRES1/?XFG2 does not contain 0 or if ?XRFNC contains ?XFOTH; otherwise, set to ?FKR0.)</p>
?XTYP (doubleword)	<p>Byte pointer to queue name (must correspond to type in ?XRFNC).</p>
?XDAT	<p>Date queued (returned by the OS in standard date notation).</p>

* There is no default unless otherwise specified.

(continued)

Table 2-16. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XXW2 (continued)	<p>For FTA queue:</p> <p>Set one or more of the following transfer option flags:</p> <ul style="list-style-type: none"> ?X2CM - compression requested. ?X2RC - process only if source more recent. ?X2AP - append source to destination if it exists. ?X2SD - delete source after transfer. ?X2DD - delete destination before transfer. ?X2RM - use record mode transfer. ?X2TO - let connection time out after failure. ?X2CP - restart at last checkpoint after failure. <p>For all other queue types: ?XXW2 equals number of copies the EXEC spooler creates. (If ?XXW2 equals 0, one copy is assumed.)</p>
?XXW2L	Reserved (Set to 0.)
?XXW3 (sometimes doubleword)	<p>For print and batch queues (doubleword): ?XXW3 contains a byte pointer to a text string (destination). This string appears in block letters at the top of any header or trailer pages. If ?XXW3 contains 0, the text string is the default username.</p> <p>However, if you supply ?PKR1 in offset ?XPRV you must supply 0 in this doubleword. The byte pointer to the text string is in offset ?XHBP of the packet extension.</p> <p>For an SNA queue, set one of these mutually exclusive flags:</p> <ul style="list-style-type: none"> ?X3TR - transparent transmission. ?X3CO - concatenate. <p>For all other queue types (single word): ?XXW3 must be 0.</p>
?XXW3L	<p>For print and batch queues: ?XXW3L contains the low-order bits of ?XXW3.</p> <p>For all other queue types: ?XXW3L must be 0.</p>

* There is no default unless otherwise specified. (continued)

?EXEC Continued

Table 2-16. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XUSR (doubleword)	For ?XRFNC equal to ?XFOTH, this field is username, otherwise it is reserved. (Set to 0.)
?XPWD (doubleword)	For ?XRFNC equal to ?FOTH, this field is password, otherwise it is reserved. (Set to 0.)
?XHBP (doubleword)	For print and batch queues ?XHBP contains a byte pointer to a text string (document name). This string appears in block letters in the middle of any header or trailer pages. If ?XHBP contains 0, the text string is the default document name.
?XMBP (doubleword)	For print and batch queues ?XMBP contains a byte pointer to a mapper filename. A mapper file alters characters, such as changing lowercase characters to uppercase characters for printing. Supply 0 if there is no mapper file.

* There is no default unless otherwise specified. (concluded)

NOTE: Supply values for the following four offsets only if offset ?XPRV contains ?PKR1.

For more information on the format of the ?XXW2 and ?XXW3 offsets, refer to the parameter files PARU.32.SR and PARU.16.SR. The sections describing the ?EXEC parameters are particularly useful when issuing queue submissions for FTA and SNA queues.

If you place ?PKR1 in offset ?XPRV and supply a local port number in offset ?XRES2/?XLPN and set Bit ?XFNC in offset ?XFGS, then ?EXEC sends an IPC message to the local port. The recipient can then notify a user that a print request is being processed.

The IPC comes from the same global port as that of an ?ILKUP of the specified queue in :PER.

All queue files in :PER have the same global port number. To identify the queue associated with the notification ?EXEC will return the queue name in the IPC message. The format of the IPC message that ?EXEC returns is in Figure 2-29.

If the operator has set the EXEC limit feature, which limits the number of pages or the CPU time allowed for your job, you must set offset ?XLMT accordingly, or default it to 0. When you default ?XLMT, EXEC sets the appropriate limit.

Offset ?XFBP is a byte pointer to the job name for batch queue requests and to the form name for other queue requests. (If you set this to 0, EXEC uses the device's previously defined default forms.) The job name, which must contain from 1 through 31 legal filename characters, identifies a batch job (one or more programs submitted as a unit to batch). The job name you select need not be unique.

?FLOCK

Locks an object.

AOS/VS

?FLOCK [*packet address*]

error return

normal return

Input

AC0 Reserved (Set to 0.)
AC1 Reserved (Set to 0.)
AC2 Address of the ?FLOCK packet, unless you specify the address as an argument to ?FLOCK

Output

AC0 Unchanged
AC1 Unchanged
AC2 Address of the ?FLOCK packet

Error Codes in AC0

ERLCK Object already locked
ERLNG Lock not granted — pended request canceled
ERLLE Lock limit exceeded — more than 1023 locks for one object
ERFNO Channel not open
ERICN Illegal channel number
ERVWP Invalid address passed as system call argument
ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

Use this system call to restrict access to objects. You can use it to prevent cooperating processes from gaining access to the same object at the same time. An object is any entity whose nature requires processes to cooperate so they can properly gain access to it. For example, an object may be a component of a file. On the other hand, an object may be a critical shared code path. Cooperating processes must agree on the meaning of the object numbers.

To achieve the desired cooperation, all processes wanting to gain access to an object must first attempt to lock the object by issuing ?FLOCK. After the operating system grants a lock on an object to a process the process may then gain access to the object. Note that the operating system does not prevent access to a locked object — it's up to the cooperating processes to make sure they don't refer to an object until the operating system has granted them a lock on the object. When a process has completed its access to an object it should release its lock by issuing system call ?FUNLOCK.

?FLOCK offers three types of lock requests: shared, exclusive, and whole file. With shared locks several processes may use an object simultaneously (e.g., reading records), but no process may gain exclusive access (perhaps for writing). Whole file locking can be either shared or exclusive. With whole-file shared locking, only whole-file shared locks are subsequently permitted. With whole-file exclusive locking, no other locks are permitted on the same file or object.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

?FLOCK Continued

What It Does

?FLOCK locks an object for use by cooperating processes. An object is locked for EXCLUSIVE or SHARED use based on the type of lock you request. To lock an object, place the number that represents the object in offset ?FLSEL of the parameter packet.

Cooperating processes use a file to represent a group of related objects that the processes want to gain access to. Objects may represent components (such as records) of the file, but this is not necessary. The file can be empty and the objects may represent entities that are separate from the file. Your process must open the file before it can lock one of its objects. Place the channel number that the open request (i.e., the ?OPEN system call) returned, along with the type of lock, in the ?FLOCK parameter packet.

Cooperating processes must have access to the same file so they can use the locking mechanism that ?FLOCK provides. Note that the operating system does *not* prevent access to a locked object — it is the responsibility of the cooperating processes not to attempt access to an object until the operating system grants them a lock on the object. You can prevent noncooperating processes from gaining access to the objects by setting the file's access control list accordingly.

For more complex situations, we suggest using a SHARED PROTECTED file (see ?SOPPF). A global server process that first opens the file controls access to a shared protected file. To prevent unauthorized access to the file when the global server process is not running, set the file's access control list to null. The global server process must then turn on its Superuser privilege so it can open the file.

To gain access to objects that the file represents, cooperating processes can issue ?RINGLD against a local server routine that then becomes a customer (see ?CON) of the global server. Upon request from a local server the global server can grant permission (see ?PMTPF) for the local server to become a subsequent opener of the file. This permission supersedes the file's access control list. The local server can then issue ?SOPPF against the file, and next use the file to represent objects by issuing ?FLOCK on the channel number that ?SOPPF returns. If these objects represent components of the file's contents, the local server can read the file by issuing ?SPAGE to gain access to an object.

Figure 2-47 shows the structure of ?FLOCK's parameter packet, and Table 2-26 describes its contents.

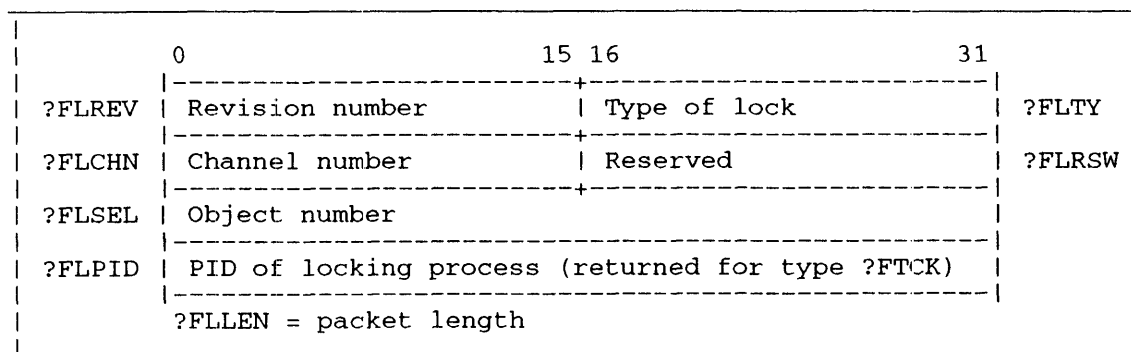


Figure 2-47. Structure of ?FLOCK Packet

Table 2-26. Contents of ?FLOCK Packet*

Offset	Contents
?FLREV	Packet revision number. Place ?PKR0 here.
?FLTY	Type of lock. It specifies the operation you want. Select from the following values. <ul style="list-style-type: none"> ?FTCK -- Check for a lock. If selected, AOS/VS does no locking but it returns the status of the object specified. If it is unlocked, then ?FLTY contains 0; if it is locked, then ?FLTY contains the lock type (?FTSH, ?FTEX, or ?FWFL). ?FTEX -- Create an exclusive lock. ?FTSH -- Create a shared lock. ?FWFL -- Lock the whole file. If you lock the whole file, you must also set either ?FTEX or ?FTSH. <p>If you supplied ?FTEX, ?FTSH, or ?FWFL, then you should also specify one of the following pending actions.</p> <ul style="list-style-type: none"> ?FTER -- Take an error return if AOS/VS is unable to lock the file element. ?FTPN -- Pend if not able to lock, and wait for an unlock to occur. AOS/VS doesn't return an error.
?FLCHN	Supply the channel number for the file.
?FLRSW	Reserved. (Set to 0.)
?FLSEL (double-word)	Object number. Your process and cooperating processes must agree on the meaning of this number.
?FLPID (double-word)	PID of the locking process. If the value of ?FTCK is set in offset ?FLTY, AOS/VS returns the PID. Otherwise, supply 0.

* There is no default unless otherwise specified.

?FLUSH

Flushes the contents of a shared page to disk.

?FLUSH

error return

normal return

Input

AC0 Any address in the page you want to flush to disk (The OS derives the logical page number by stripping off Bits 22 through 31.)

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Undefined

AC2 Undefined

Error Codes in AC0

ERNSA Shared I/O request not to shared area

ER_FS_DIRECTORY_NOT_AVAILABLE

Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

?FLUSH directs the operating system to write a shared page out to disk immediately. Therefore, ?FLUSH is useful if you have modified a shared page and you want to make sure that the operating system updates it immediately.

Note that before you can use ?FLUSH, you must use ?SOPEN to open the target file for shared access.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?FLUSH copies the contents of a shared page from memory to disk, without actually releasing the shared page from memory. The calling process regains control after the operating system has performed the flush.

Notes

- See the description of ?RPAGE and ?ESFF in this chapter.
- See the descriptions of ?SOPEN and ?UPDATE in this chapter for information on simulating ?FLUSH.

?FSTAT

Gets file status information.

?FSTAT [*file status packet address*]

error return

normal return

Input

AC0 may contain one of the following:

- Byte pointer to the pathname of the target file
- Channel number of the target file

AC1 Flags:

Bit 0 = 0 if AC0 contains a byte pointer
Bit 0 = 1 if AC0 contains a channel number
Bit 1 = 0 to resolve links in the pathname
Bit 1 = 1 to ignore links in the pathname

AC2 Address of the file status packet, unless you specify the address as an argument to ?FSTAT

Output

AC0 Unchanged

AC1 Unchanged

AC2 Address of file status packet

Error Codes in AC0

ERCIU Channel in use
ERFAD File access denied
ERFNO Channel not open
ERICN Illegal channel
ERMPR System call parameter address error
ERVBP Invalid byte pointer passed as a system call argument
ERVWP Invalid word pointer passed as a system call argument
ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)

?FSTAT Continued

Why Use It?

?FSTAT lets you determine the specifications set for a file or a directory when it was created. ?FSTAT also returns the date and time the file was last accessed or modified. (You may need this information before you open or reopen a file.)

Who Can Use It?

There are no special process privileges needed to issue this call. See the following section “What It Does” for an explanation of file access requirements.

What It Does

?FSTAT returns the status parameters of the target file. You can specify the packet address as an argument to ?FSTAT or you can load the address into AC2 before you issue ?FSTAT. Before the calling process can issue ?FSTAT, it must meet these rules:

- If it has no access rights to a file, it needs Read and Execute access to the file's directory.
- If it has any access right to a file, it needs Execute access to the file's directory.

Before you issue ?FSTAT, load AC0 with either the channel number of the target file or a byte pointer to its name.

If you specify a pathname that ends in a link and Bit 1 of AC1 equals 1, the operating system returns status information about the link without resolving it. If the pathname ends in a link and you do not set Bit 1 of AC1, the operating system resolves the link entry and returns status information about the file to which the link refers.

The format of the information ?FSTAT returns varies, depending on the type of file. The operating system returns different information packets for IPC files, directories, unit files, and other file types. Unit files are devices that have been opened as a unit (for example, MCAs, MTBs, disks, etc.). Figure 2-48, Figure 2-49, Figure 2-50, and Figure 2-51 show the structure of the various ?FSTAT packets.

?FTOD

Converts time of day to a scalar value.

?FTOD

error return

normal return

Input

AC0 Seconds from 0 through 59

AC1 Minutes from 0 through 59

AC2 Hour from 0 (midnight)
through 23 (11 p.m.)

Output

AC0 Number of biseconds (seconds/2)
since midnight

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERPRE Invalid system call parameter

Why Use It?

You can use ?FTOD to obtain input parameters for the ?CREATE system call's time block.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?FTOD takes time in seconds, minutes, and hours and converts it to a scalar value; that is, the number of biseconds (half the number of seconds) since midnight.

Before you issue ?FTOD, load AC0, AC1, and AC2 with the appropriate octal values for seconds, minutes, and hours, respectively. The legal input parameters for ?FTOD range from 0:00:00 (midnight) to 23:59:59 (11:59 p.m.).

The operating system returns the scalar time value to AC0. If the scalar conversion results in an odd number of seconds, the operating system rounds the value to the next number before division; for example, 7 seconds would result in $(8/2) = 4$ biseconds.

?FUNLOCK

Unlocks an object.

AOS/VS

?FUNLOCK [*packet address*]

error return

normal return

Input

AC0 Reserved (Set to 0.)
AC1 Reserved (Set to 0.)
AC2 Address of the ?FUNLOCK packet, unless you specify the address as an argument to ?FUNLOCK

Output

AC0 Unchanged
AC1 Unchanged
AC2 Address of the ?FUNLOCK packet

Error Codes in AC0

ERNLK Object not locked
ERVWP Invalid address passed as system call argument
ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

Use this system call to unlock an object when you no longer need it.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?FUNLOCK releases a lock that your process has placed.

?FUNLOCK lets other processes gain a lock on the same object. It can also unlock all locks that your process holds on the specified file. You must have issued ?OPEN for a file before you can unlock one of its objects. Place the channel number ?OPEN returns in the ?FUNLOCK packet.

Cooperating processes must have access to the same file in order to use the unlocking mechanism that ?FUNLOCK provides. The processes must also agree on the meaning of the object numbers so they can guarantee successful unlocking.

Figure 2-53 shows the structure of ?FUNLOCK's parameter packet, and Table 2-28 describes its contents.

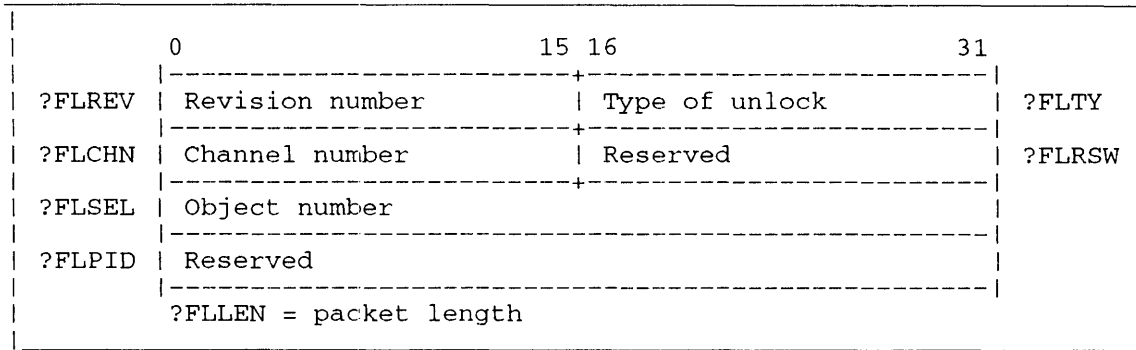


Figure 2-53. Structure of ?FUNLOCK Packet

Table 2-28. Contents of ?FUNLOCK Packet*

Offset	Contents
?FLREV	Packet revision number. Place ?PKR0 here.
?FLTY	Type of unlock. It specifies the operation you want. Select from the following values. ?FULA --- Unlocks all locks that the process holds on this channel. If you don't select ?FULA, AOS/VS unlocks only the lock specified by offset ?FLSEL. ?FWFL --- Unlocks the whole file.
?FLCHN	Supply the channel number for the file.
?FLRSW	Reserved. (Set to 0.)
?FLSEL (double-word)	Object number. Your process and cooperating processes agree on the meaning of this number.
?FLPID (double-word)	Reserved. (Set to 0.) ?FLOCK uses this offset.

* There is no default unless otherwise specified.

?GACL

Gets a file entry's access control list (ACL).

?GACL [*packet address*]

error return

normal return

Operating System Differences

AOS/RT32 always returns "+,OWARE" as the ACL.

Input

AC0 One of the following:

- Byte pointer to the pathname of the target file
- 0 if a packet address is supplied

AC1 Byte pointer to a receive buffer for the ACL

AC2 One of the following:

- Reserved (Set to 0 if not supplying a packet.)
- Address of the ?GACL packet, unless you specify the address as an argument to ?GACL

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERRAD Read access denied

ERVBP Invalid byte pointer passed as a system call argument

ER_FS_DIRECTORY_NOT_AVAILABLE

Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

Because ?GACL returns a file or directory's current ACL, you can use it along with ?SACL, which sets the current ACL, or with ?DACL, which sets the default ACL.

Who Can Use It?

There are no special process privileges needed to issue this call. If you specified the file with a channel number, your process must have Read or Write access to the target entry's parent directory or Owner access to the target entry. If you specified the file with a pathname, you must also have Execute access to the target entry's parent directory.

What It Does

?GACL returns the access control list (ACL) associated with a file or directory. AOS/RT32 always returns "+,OWARE" as the ACL.

?GLIST

Gets the contents of a search list.

?GLIST

error return

normal return

Input

AC0 Reserved (Set to 0.)
AC1 Byte pointer to a receive
buffer for the search list
AC2 Byte length of the receive
buffer

Output

AC0 Undefined
AC1 Unchanged
AC2 Unchanged

Error Codes in AC0

ERVBP Invalid byte pointer passed as a system call argument

ER_FS_DIRECTORY_NOT_AVAILABLE

Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

?GLIST lets you examine the current contents of your search list. Thus, you can use ?GLIST as a preliminary system call to ?SLIST, which changes the contents of the search list.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?GLIST returns a copy of the calling process's search list to the buffer that you specify in AC1.

Before you issue ?GLIST, load AC1 with a byte pointer to the receive buffer and load AC2 with the length of the buffer. (The symbol ?MXPL represents the maximum length of the search list buffer.) Then, after you issue ?GLIST, the operating system returns the search list to the buffer in the following format:

```
pathname<0>...pathname<0><0>
```

where

pathname is a complete pathname string.

<0> is the null terminator.

Be sure to allocate enough buffer space to accommodate the two trailing nulls that the operating system appends to the search list.

Notes

- See the description of ?SLIST in this chapter.

?GMEM

Returns the number of undedicated memory pages.

?GMEM

error return

normal return

Input

None

Output

AC0 Undefined

AC1 Current number of undedicated memory pages

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?GMEM gives you information about overall memory contention. In general, the smaller the value returned by ?GMEM, the more likely memory contention becomes. Issuing ?GMEM before you wire pages to a resident process's working set helps you to correctly adjust the size of the resident process to your memory configuration. Do not wire more pages than the current number of undedicated memory pages, because it can cause a system deadlock.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?GMEM returns the current number of undedicated pages available to the calling process. Undedicated pages are unused pages that the operating system can allocate to the calling process.

Notes

- See the description of ?WIRE in this chapter.

?GNAME

Gets a complete pathname.

?GNAME

error return

normal return

Input

AC0 Byte pointer to a filename
or pathname fragment

AC1 Byte pointer to a receive
buffer for the pathname

AC2 Byte length of the receive
buffer

Output

AC0 Unchanged

AC1 Unchanged

AC2 Byte length of the complete
pathname, excluding the null terminator

Error Codes in AC0

ERFAD File access denied

ERFDE File does not exist

ERIRB Insufficient room in buffer

ERMPR System call parameter address error

ERVBP Invalid byte pointer passed as a system call argument

Why Use It?

?GNAME can be useful as a preliminary system call for system calls that require a complete pathname as input.

Who Can Use It?

Required file access privileges depend on whether your program runs under the old file system (before AOS/VS II Release 1.00 and with any revision of AOS/RT32 or of AOS/VS) or under the new file system (effective with AOS/VS II Release 1.00).

Old File System

There are no special process privileges needed to issue this call. The calling process must have access (of any type) to each file in the pathname fragment or have Read access to their parent directories. The calling process must also have Execute access to the files' parent directories. However, the operating system does not check for access privileges if the pathname fragment consists of only a prefix (for example, =).

New File System

There are no special process privileges needed to issue this call. The calling process must have access (of any type) to each file in the pathname fragment or have Read access to their parent directories. The calling process must also have Execute access to the files' parent directories. However, the operating system does check for access privileges in all cases, including when the pathname fragment consists of only a prefix (for example, =). In this case, the calling process still must have access (of any type) to each file in the pathname fragment or have Read access to their parent directories.

?GNAME Continued

An Important Difference

One important difference between issuing ?GNAME under the old and new file systems occurs in the following sequence of events:

- The CLI issues ?PROC to create a new child CLI process.
- The newly created child CLI process issues ?SUSER and ?SUPROC to explicitly disable Superuser and Superprocess mode in case the parent process has either privilege.
- The child CLI process issues ?GNAME against “=” to determine the full pathname of the initial directory. Furthermore, suppose the child process does not have access to its working directory, “=”.
 - Under the old file system ?GNAME succeeds because AOS/VS performs no access checking; however, the child CLI process has no access to its initial directory. If this process changes its working directory and tries to reenter the initial directory (via, say, a **DIR/I** command), ?GNAME returns the error ERFAD.
 - Under the new file system ?GNAME fails right away with error ERFAD. The child CLI process terminates.

What It Does

?GNAME returns the complete pathname, starting with the root, when you specify a pathname fragment in AC0. ?GNAME functions with either an open or a closed file. If you load AC0 with the byte pointer to the prefix =, the operating system returns the pathname of the current working directory.

You cannot use ?GNAME to obtain the “true” pathname associated with a generic file. For example, ?GNAME would return a complete pathname of :PER:DATA for the input pathname @DATA, even though the “true” complete pathname is really :UDD:USER:DATA. Therefore, to obtain the “true” pathname, use ?GRNAME.

Notes

- See the description of ?GRNAME in this chapter.

?GNFN

Lists a particular directory's entries.

?GNFN [*packet address*]

error return

normal return

Input

AC0	Reserved (Set to 0.)
AC1	Channel number of the target directory
AC2	Address of the ?GNFN packet, unless you specify the address as an argument to ?GNFN

Output

AC0	Undefined
AC1	Unchanged
AC2	Address of the ?GNFN packet

Error Codes in AC0

ERCIU	Channel in use
EREOF	End of file (There are no more files in the target directory.)
ERFNO	Channel not open
ERFTL	Filename too long (The template exceeds 63 characters.)
ERIFC	Illegal filename character
ERNDR	Not a directory (The channel that you specified is not opened on a directory.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument
ER_FS_DIRECTORY_NOT_AVAILABLE	Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

?GNFN allows you to obtain a specific directory's filenames. The CLI, for example, uses ?GNFN to implement its FILESTATUS command.

Who Can Use It?

There are no special process privileges needed to issue this call. However, your process must be able to open (via ?OPEN or ?GOPEN) the directory. It must also have Read and Execute access to the directory at the time it issues ?OPEN or ?GOPEN.

What It Does

?GNFN returns a filename in the target directory (or all filenames that match the template that you specify) to the buffer that you specify in the ?GNFN packet. The first time you issue ?GNFN, the operating system returns the name of the first file in the directory; the second time, it returns the second filename, and so on.

To get all the filenames, use the + template. You can also obtain every filename by issuing repeated ?GNFN system calls until ?GNFN fails on error code EREOF. (EREOF is an end-of-file condition that means you have reached the last file entry in the directory.)

?GNFN Continued

Before you issue ?GNFN, you must open the target directory. After you open the directory with ?OPEN (or ?GOPEN), load AC1 with the correct channel number, which returns in the ?OPEN (or ?GOPEN) packet. You can specify the packet address as an argument to ?GNFN, or you can load it into AC2 before you issue ?GNFN. Figure 2-55 shows the structure of the ?GNFN packet, and Table 2-34 describes its contents.

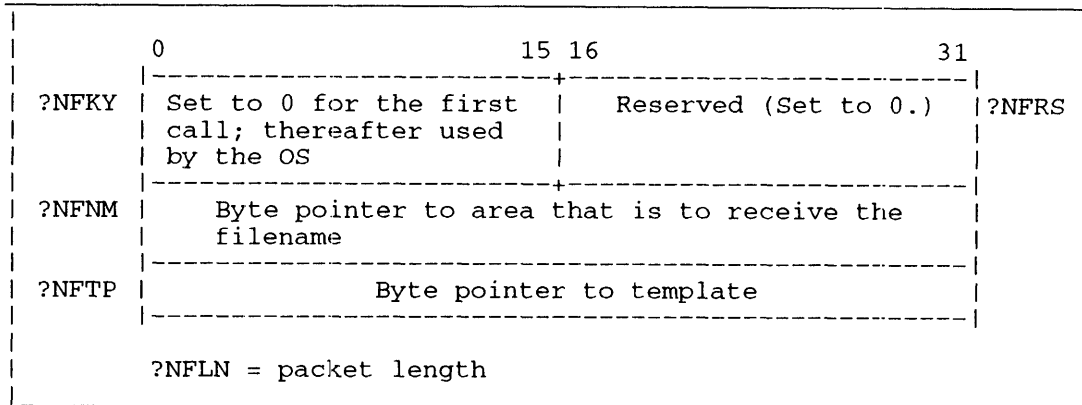


Figure 2-55. Structure of ?GNFN Packet

Table 2-34. Contents of ?GNFN Packet*

Offset	Contents
?NFKY	OS internal value. Set to 0 the first time you issue ?GNFN; ignore thereafter. This value is a key to where the list should continue. So do not set ?NFKY to zero between calls.
?NFRS	Reserved. (Set to 0.)
?NFM (doubleword)	Byte pointer to area that is to receive the filename.
?NFTP (doubleword)	Byte pointer to filename template, if present. DEFAULT = -1 (no template). If you omit the template, the operating system searches only the open directory. Table 2-35 shows the template characters.

* There is no default unless otherwise specified.

Table 2-35. Filename Template Characters

Character	Description
* (asterisk)	Matches any single filename character except a period.
- (hyphen)	Matches any series of characters not containing a period.
+ (plus sign)	Matches any series of characters.

The first time you issue ?GNFN, set the contents of offset ?NFKY to 0. Do not modify the contents of ?NFKY on subsequent system calls, because the operating system uses this offset as an internal pointer. There is no default value for offset ?NFM. Therefore, you must use offset ?NFM as a byte pointer to a receiving area for the filename. If you do not want to use a template, choose the default value (-1) of offset ?NFTP.

Sample Packet

```
PKT:   .BLK    ?NFLN           ;Allocate enough space for the packet
                                     ;(packet length is ?NFLN).
       .LOC    PKT+?NFKY       ;Set to 0 for first call. The OS
       .WORD   0               ;uses this value for second and
                                     ;subsequent calls.
       .LOC    PKT+?NFM        ;Byte pointer to an area to receive
                                     ;the filename.
       .DWORD  BUFF*2          ;Byte pointer to BUFF.
       .LOC    PKT+?NFTP       ;Byte pointer to template.
       .DWORD  -1              ;No template (default is -1).
       .LOC    PKT+?NFLN       ;End of packet.
```

Notes

- See the description of ?OPEN in this chapter.

?GOPEN

Opens a file for block I/O.

?GOPEN [*packet address*]

error return

normal return

Operating System Differences

Only AOS/VS II supports ?ODTL in offset ?ODF1 of the packet extension.

Input

AC0 Byte pointer to the target
file's pathname

AC1 Target file's channel
number, or -1, to direct the
OS to assign a channel number

AC2 Address of the ?GOPEN
packet, unless you specify the
address as an argument to
?GOPEN

Output

AC0 Contains the following:

- Bits 0 through 25 are reserved (Set to 0.)
- Bits 26 through 31 contain one or more of the following file access privilege masks:

?FACO	Owner access
?FACW	Write access
?FACA	Append access
?FACR	Read access
?FACE	Execute access

AC1 Unchanged

AC2 Address of the ?GOPEN packet

Error Codes in AC0

ERDMO Disk marked as owned by another system

ERDRS Device reserved by another port

ERFTM File/tape mismatch

ERITD Indecipherable tape density

ERVBP Invalid byte pointer passed as a system call argument

ERVWP Invalid word pointer passed as a system call argument

ER_FS_NO_TLA_PRIVILEGE

Attempt to issue ?GOPEN with ?ODTL value supplied, but without proper privilege

ER_FS_OBSOLETE_IPC_FILE_DETECTED

Obsolete IPC file type has been detected; file has been deleted (AOS/VS II only)

Why Use It?

You must use ?GOPEN to open a file for block I/O or physical block I/O. Also, you must issue ?GOPEN to open an MCA as a unit or to open a specific link. Then, if you use the CLI MOUNT command to mount a tape, you must ?GOPEN the file using its link name.

Who Can Use It?

To issue this call, you need a special privilege for only one situation. Your process must be running under the new file system (AOS/VS II) and in Superuser mode in order to issue ?GOPEN with value ?ODTL supplied in offset ?ODF1 of the packet extension.

You must have Read and Execute access to the file's parent directory. In addition, you must have Read access to the file to read from it and Write access to the file to write to it.

What It Does

?GOPEN opens a file for block I/O and assigns the file a channel number. You can specify the channel number in AC1 or force the operating system to assign the channel number by loading AC1 with -1. You can open any of the following devices for block I/O: disk files and units, tape file, MTB tape units, MTD tape units, and MCA units.

The ?GOPEN packets are different for IPC files (see Figure 2-56 and Table 2-36) and for all other file types (see Figure 2-57 and Table 2-38). Although you must reserve sufficient space in your logical address space for the packets, you need to provide input specifications for offset ?OPFL (for the standard packet) or offset ?OPCH (for the IPC packet) only.

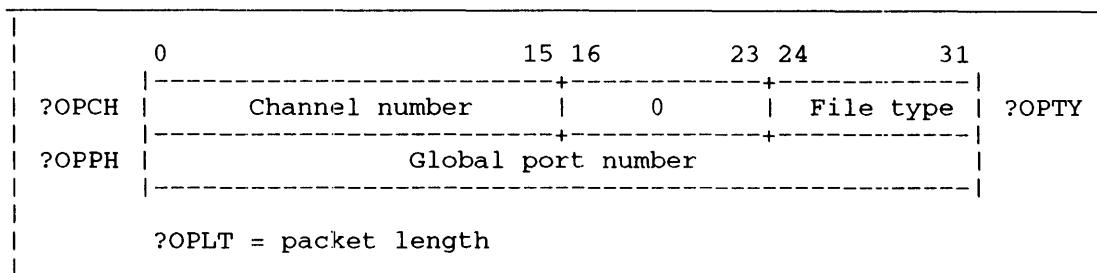


Figure 2-56. Structure of ?GOPEN Packet for IPC Files

Table 2-36. Contents of ?GOPEN Packet for IPC Files*

Offset	Input Value	Output Value
?OPCH	None.	Channel number.
?OPTY	None.	Left byte: 0 Right byte: File entry type (?FIPC)
?OPPH (doubleword)	None.	Global port number.

* There is no default unless otherwise specified.

?GOPEN Continued

Table 2-37. Contents of Standard ?GOPEN Packet*

Offset	Input Value	Output Value
?OPFL	Flag bits ?OPME--Exclusive open. ?OPMD--Inhibit initial form feed. ?OPXP--Packet has extension. If tape density mode, specify one of the following: ?OPDL--800 bpi. ?OPDM--1600 bpi. ?OPDH--6250 bpi. ?OPAM--automatic density mode matching selected ?OPD5--Low density ?OPD6--Medium density ?OPD7--High density Transfer mode (for Model 6352 magnetic tape units only): ?OPMBF--Use buffered mode when per- forming tape I/O ?OPMST--Use streaming mode when per- forming tape I/O DEFAULT--0 for non- buffered and non-streaming data transfer	Bits 0-15: Channel number.
?OPTY	None.	Left byte: Record format Right byte: File entry type
?OPFC	None.	File control parameters. See the description of offset ?SCPS in the explanation of ?FSTAT.
?OPEW	None. (Set to 0.)	Unchanged.
?OPEH (doubleword)	None	File size in bytes.

* There is no default unless otherwise specified.

The operating system returns all the other parameters, based on the specifications you set when you created the file. Although the operating system returns file type and format information, the block I/O and physical block I/O system calls ignore record formatting information, because they operate in terms of blocks only.

Notice that offset ?OPFL/?OPCH in both packets contains different input and output values. On input, you can use ?OPFL/?OPCH to indicate a tape-density mode if you are opening a magnetic tape file on a type MTB or MTD controller. On output, ?OPFL/?OPCH contains the target file's channel number.

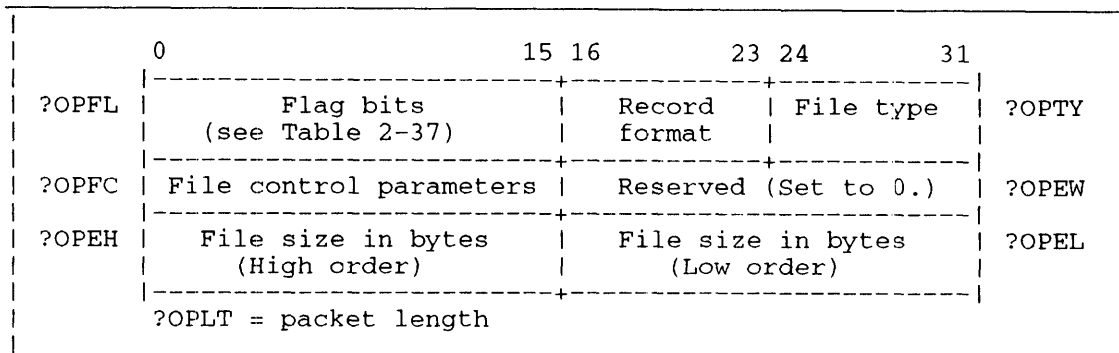


Figure 2-57. Structure of Standard ?GOPEN Packet

Opening System Areas

You can issue ?GOPEN against a system area. To do this, simply place a byte pointer to the desired unit name and system area number in AC0. An example of a system area name is @DPJ0:1082. You must have previously created, via system utility Disk Jockey, the system area. You can open a system area only under the New File System.

Issuing ?GOPEN against a system area is much the same as issuing ?GOPEN against a physical disk alone, but in the first case you can gain access to only part of the disk. Bad block remapping does not occur for system areas that you issue ?GOPEN against.

?GOPEN Options

To open the file to the calling process exclusively, supply value ?OPME in offsets ?OPFL/?OPCH at input. Note that this option is available only within the standard packet (for non-IPC files), because it makes no sense to exclusively open an IPC file.

If you are reading or writing a magnetic tape file on an MTB or MTD tape controller, you can select a density mode for the tape controller by supplying one of the density-mode values in offsets ?OPFL/?OPCH.

Supply ?OPDL to specify a density of 800 bpi (bytes per inch); supply ?OPDM to set the density to 1600 bpi; supply ?OPDH to specify 6250 bpi; or supply ?OPAM if you want the operating system to select the density mode automatically.

Supply ?OPD5, ?OPD6, and ?OPD7 for low, medium, or high densities respectively. When you specify low or high density, the OS selects the lowest or highest density supported by the drive. When you select medium density: if the drive supports three levels, the OS selects the middle density value; when the drive supports two densities, the OS selects the lower density.

If you have issued ?GOPEN with value ?ODTL supplied, you cannot issue the following system calls to the file:

- ?ALLOCATE
- ?BLKIO (write block only)
- ?CPMAX
- ?CRUDA
- ?DELETE
- ?ESFF
- ?GTRUNCATE
- ?PWRB
- ?SACL
- ?UPDATE
- ?WRB
- ?WRUDA

An attempt to issue one of these system calls with value ?ODTL supplied in the ?GOPEN packet results in error code ER_FS_TLA_MODIFY_VIOLATION.

?GOPEN Continued

When you default the density-mode parameter (that is, omit the values), the operating system uses the density mode you chose for the MTB or MTD controller during the system-generation procedure. Make sure the default density mode matches that of your tape; otherwise, the operating system fails on error code ERFTM (file tape density mismatch). However, density matching always occurs for reads on MTD drives unless you specify a density.

If you select ?OPAM (automatic density selection) and the tape unit is damaged or the operating system cannot set the density, ?GOPEN fails on error code ERITD (tape density indecipherable).

In buffered mode (Model 6352 magnetic tape units only) the tape controller indicates that a tape transfer is complete after data has been read from memory, but before it has been written to tape. In this mode the system might not report error conditions for a request that fails. Therefore your program must check for errors whenever it issues a ?GCLOSE call. (The system reports all errors when it writes a file mark. Since ?GCLOSE writes a file mark, any program using buffered mode must explicitly close each tape file.)

The streaming mode (Model 6352 magnetic tape units only) allows your program to open a tape unit for high-speed backup purposes. In this mode the tape controller expects data transfer with the tape to occur quite rapidly. If data is not transferred fast enough, the performance of the tape unit degrades. So, a program that cannot maintain a high data transfer rate should not select this mode.

?GOPEN Extension

The ?GOPEN extension is used to allow access to options that exist for some Data General disk subsystems. These options (described below) may not be supported by all disk subsystems. An error will be returned (typically ERIOD or ERMNS) if the option selected is not available for the specified device.

The extension allows access to the following hardware features:

- Format selection for 4514 (DPM) disks.
- Modified sector I/O for DPJ disks (only large capacity models).
- Trespass option for dual ported disks.
- Model option for some MTJ tapes (Model 6352 only).
- Tape drives supporting data compression and the SCSI-2 protocol.

Supply value ?ODTL in offset ?ODF1 of the packet extension when you do not want the operating system to change the file's date/time last accessed values. These values are offsets ?STAH and ?STAL of the parameter packet for system call ?FSTAT and are also the values that the /TLA switch of the CLI FILESTATUS command returns. This change of date/time values occurs by default (i.e., ?ODTL not supplied) when your process later reads or closes the file; it will not occur if you have supplied ?ODTL.

Data compression is on (the default condition) for tape drives supporting compression. You must switch it off if you do not want to use it. Set ?ODCOF to 1 in offset ?ODF1 of the packet extension when you want to turn compression off and operate the drive in native mode.

To use the extension, you must set the extension bit (?OPXP) in the flag word (?OPFL) of the ?GOPEN packet. Figure 2-58 shows the structure of the ?GOPEN packet extension and Table 2-38 contains the option flags for offset ?ODF1 of the ?GOPEN packet extension. The number of words in the combined ?GOPEN main packet and its extension is ?OPLT + ?OPXL.

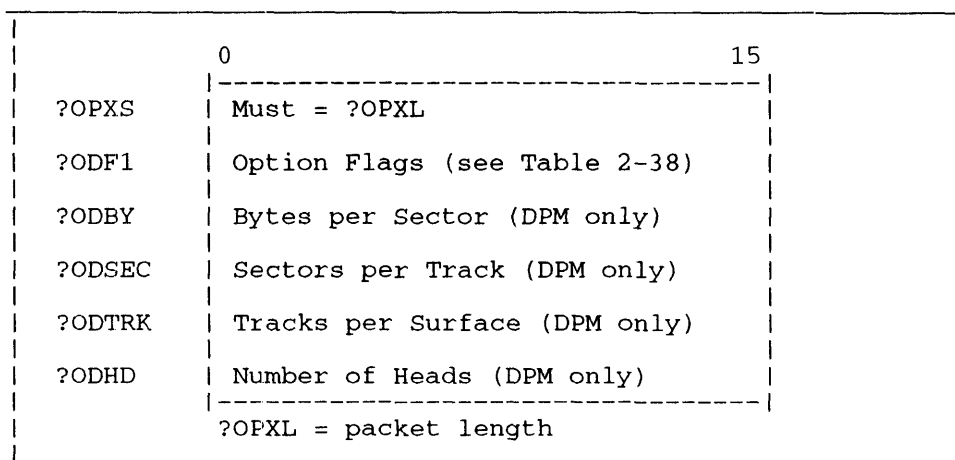


Figure 2-58. Structure of ?GOPEN Packet Extension.

Table 2-38. Option Flags for Offset ?ODF1

Flag	Description
?ODND	Write without Modified bits (DPJ only)
?ODMB	Allow Modified Sector I/O (DPJ only)
?ODHS	Swap Head Option (DPM only)
?ODBS	Swap Byte Option (DPM only)
?ODP0	First Physical Sector is 0 (DPM only) (Used for Enterprise/MPT format only)
?ODST	Choose streaming mode for tape (MTJ only)
?ODTEO	9-track tape emulation override
?ODCOF	Data compression off If ?ODCOF = 0 the tape operates in data compression mode. If ?ODCOF = 1, then data compression is turned off, and the drive operates in native mode.
?ODTP	Old file system -- trespass if reserved by another port (DPJ only); new file system -- trespass if reserved by another port or if marked as owned by another system
?ODTL	Do not change the time last accessed value when the file is read from or closed.

?GOPEN Continued

Table 2–39 contains the specific format selectors that Figure 2–58 introduces.

Table 2–39. Valid Format Options (DPM disks)

Offset	8 Sector Format	9 Sector Format (Standard AOS)	10 Sector Format (Enterprise/MPT)
?OPXS	?OPXL	?OPXL	?OPXL
?ODF1	0	0	?ODHS+?ODBS+?ODP0
?ODBY	512.	512.	512.
?ODSEC	8.	9.	10.
?ODTRK	40.	40.	35.
?ODHD	1 or 2	1 or 2	2

In order to perform modified sector I/O (DPJ disks) using the ?BLKIO call, bit ?ODMB must be set. If this bit is not set, an error (ERCPO) will be returned. Use the ?ODND option to clear modified bits in conjunction with the ?BLKIO call.

Sample Packet

The following sample packet shows the standard ?GOPEN packet:

```
PKT:   .BLK   ?OPLT           ;Allocate enough space for the
      ;packet. Packet length = ?OPLT

      .LOC   PKT+?OPFL       ;Channel number.
      .WORD  0               ;The OS returns this value.

      .LOC   PKT+?OPTY       ;Record format and file type.
      .WORD  0               ;The OS returns this value.

      .LOC   PKT+?OPFC       ;Record length (if fixed format).
      .WORD  0               ;The OS returns this value.

      .LOC   PKT+?OPEW       ;Reserved.
      .DWORD 0               ;You must set this value to 0.

      .LOC   PKT+?OPEH       ;File size (in bytes).
      .DWORD 0               ;The OS returns this value.

      .LOC   PKT+?OPLT       ;End of packet.
```

Notes

- See the description of ?GCLOSE in this chapter.
- Refer to the chapter about VSGEN in the manual *Installing, Starting, and Stopping AOS/VS II* for information on the system-generation procedure.

?GTRUNCATE

Truncates a disk file.

?GTRUNCATE [*packet address*]

error return

normal return

Input

AC0 Reserved (Set to 0.)

AC1 Channel number of the disk file

AC2 Address of the ?GTRUNCATE packet, unless you specify the address as an argument to ?GTRUNCATE.

Output

AC0 Undefined

AC1 Unchanged

AC2 Address of the ?GTRUNCATE packet

Error Codes in AC0

EREOF End of file (You tried to set the EOF (end-of-file) past the current EOF.)

ERFNO Channel not open

ERMUS Multiple users of file; cannot truncate

ERSHR Shared file; cannot truncate

ERSIM Simultaneous requests on same channel

ERVWP Invalid word pointer passed as a system call argument

ER_FS_DIRECTORY_NOT_AVAILABLE
Directory not available because the LDU was force released (AOS/VS II only)

ER_FS_TLA_MODIFY_VIOLATION
Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet

Why Use It?

You can use ?GTRUNCATE to reduce the size of a disk file that is currently open for block I/O. Thus, ?GTRUNCATE allows you to reclaim disk space that you no longer need.

Who Can Use It?

There are no special process privileges needed to issue this call. You must have access to the file's channel number (via ?OPEN or ?GOPEN) and you must have Write access to the file.

What It Does

?GTRUNCATE truncates (shortens) disk files. The caller passes a channel number and a pointer to a new ?GTRUNCATE packet. The packet contains the new size of the file in bytes. This new size must be less than or equal to the current size of the file or the operating system returns error code EREOF (end of file).

If you need to truncate a magnetic tape file, use ?TRUNCATE.

The following restrictions apply to files that you want to truncate:

- The file must have a use count of +1.

?GTRUNCATE Continued

- The file must not be open for shared I/O.
- The file should have been opened with ?GOPEN instead of ?OPEN.

Figure 2–75 shows the structure of the ?GTRUNCATE packet, and Table 2–55 describes its contents.

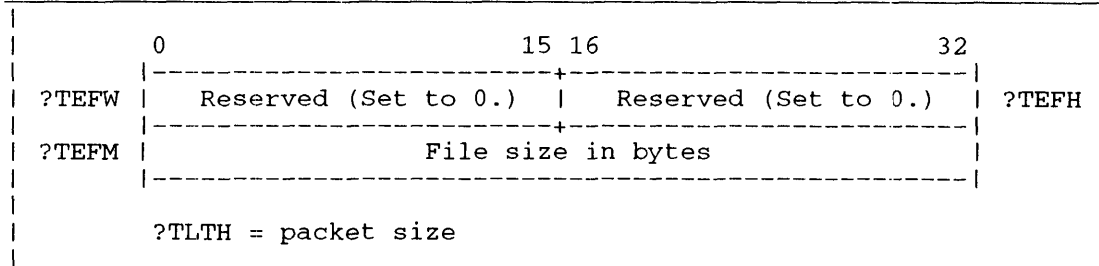


Figure 2–75. Structure of ?GTRUNCATE Packet

Table 2–55. Contents of ?GTRUNCATE Packet*

Offset	Contents
?TEFW	Reserved. (Set to 0.)
?TEFH	Reserved. (Set to 0.)
?TEFM (doubleword)	New file size in bytes. The new file size must be less than or equal to the current (old) file size.

* There is no default unless otherwise specified.

When you use ?GTRUNCATE, be sure to reset the values of offset ?PRNH and ?PRCL in the block I/O packet.

Notes

- See the descriptions of ?OPEN and ?GOPEN in this chapter.
- See the descriptions of ?PRDB/?PWRB and ?BLKIO in this chapter for information on the block I/O packet.

?IDEF

Defines a user device.

?IDEF

error return

normal return

Input

Output

AC0	Contains the following: <ul style="list-style-type: none">• Bit 1 is a flag bit: Bit 1 = 0 if AC2 specifies the number of data channel map slots needed (no map definition table) Bit 1 = 1 if AC2 points to a data channel map definition table• Bit 2 is a flag bit for 16 bit processes: Bit 2 = 1 if you want to use the extended packet• Bits 3 through 31 contain the device code for the user-defined device in the range from 1 through 191 (= 277 octal)	AC0	Unchanged
AC1	Bits 1 through 31 contain the address of the device's DCT; set Bit 0 to 1 if the device is a data channel (DCH) or burst multiplexor channel (BMC) device	AC1	Unchanged
AC2	One of the following: <ul style="list-style-type: none">• Address of the map definition table• Number of map slots needed (no map definition table) (each map slot accesses 1K words)	AC2	Unchanged

If your program executes as a 16-bit process, then the bit numbers in AC0 and AC1 change from above as follows:

Bit 0	would read Bit 16
Bit 1	would read Bit 17
Bits 1 through 31	would read Bits 17 through 31
Bits 2 through 31	would read Bits 18 through 31

?IDEF Continued

Error Codes in AC0

ERDCH	Data channel map full
ERDNM	Illegal device code (The device code is outside the legal range (1 through 191).)
ERIBS	Device already in use
ERPRE	Invalid system call parameter
ERPRV	Caller is not privileged for this action
ERPTY	Illegal process type

Why Use It?

?IDEF lets you establish an interface between the operating system and a device it does not support. ?IDEF and the other user-device system calls are particularly useful if you have applications-specific peripheral devices for which you have written special device-driver routines.

Who Can Use It?

The caller must be a resident process and must have privilege ?PVDV to use ?IDEF. There are no restrictions concerning file access.

What It Does

?IDEF defines a user device and its device control table (DCT). The operating system builds an internal DCT based on your DCT specifications, and enters this into its interrupt vector table.

Before you issue ?IDEF, set up a device control table in your logical address space and load its address into Bits 1 through 31 of AC1. (See Figure 2-77 for the DCT format if you want to issue ?IDEF from a 32-bit process. See Figure 2-78 if you want to issue ?IDEF from a 16-bit process.) Be sure to set offset ?UDVIS in the DCT to the address of the interrupt service routine for the new device, and define the interrupt service mask in offset ?UDVMS. In addition, you must set offset ?UDVBX (the mailbox) to 0.

For devices that reside on the secondary IOCs (i.e., device codes 64.-191.), you must use PIO instructions to communicate with your device. NOVA I/O is limited to the first IOC.

You may extend the DCT to include a word pointer to a device termination routine. In this case control passes to the routine if your process traps or terminates. This transfer of control prevents a runaway ?IDEF-specified device from altering system databases. Also, the environment would be the same at process termination time as if the device had just requested an interrupt. You might specify the extended DCT if you are writing an interrupt service routine (ISR) for a device such as an intelligent asynchronous controller (IAC) or an intelligent synchronous controller (ISC).

To extend the DCT, place ?UDLX in offset ?UDRS and place a word pointer to your device termination routine in offset ?UDDTR. The length of the extended packet is ?UDLE (= ?UDLN+?UDLX) words.

To specify the standard DCT, place 0 in offset ?UDRS and allow ?UDLN words for the unextended packet. In other words, programs written before you could extend the DCT (effective with AOS/VS Revision 7.00) will not have to change.

?IDEF Continued

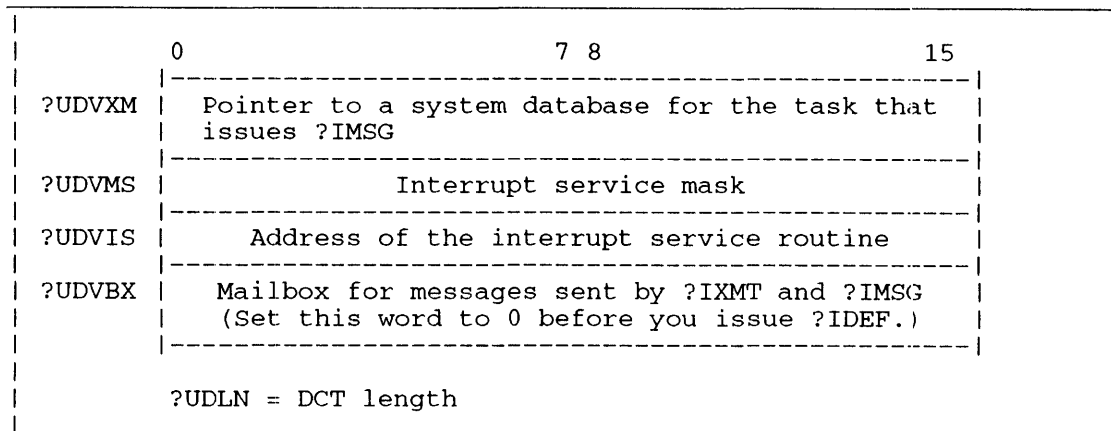


Figure 2-78. Structure of Device Control Table (DCT) for 16-Bit Processes

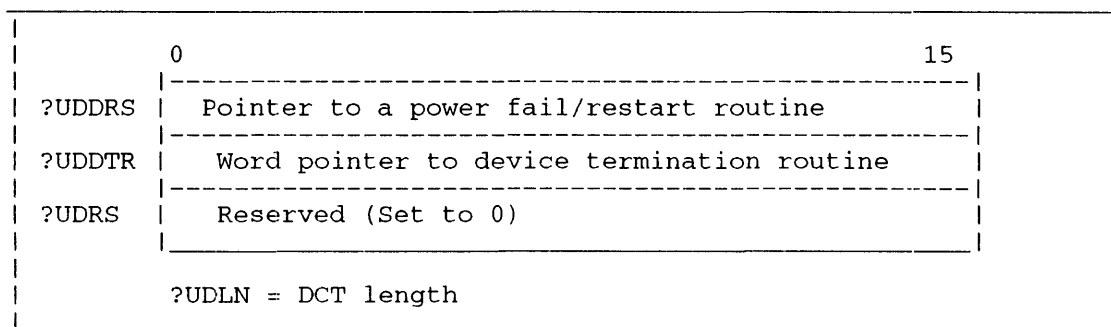


Figure 2-78.1. Structure of Extended Packet for 16-Bit Processes

Options

Set Bit 0 of AC1 if you want to use either the data channel (DCH) or the burst multiplexor channel (BMC) for I/O transfers to and from the new device. If you choose the DCH or BMC option, you must also define the number of map slots the device will need. You can load the map slot value into AC2 before you issue ?IDEF or you can set up a map definition table in your address space. If you set up a map definition table, load its address into AC2 before you issue ?IDEF. (If you use AC2, the operating system will allocate map slots in DCH map A.)

Set Bit 2 of AC0, if you want to use the extended packet to clear LBUS interrupts. Figure 2-78.1 shows the structure of the extended packet.

The map definition table specifies the first acceptable map slot for BMC or DCH transfers, and optionally, selects a particular DCH map (maps A through P). The map definition table can contain as many as eight entries. Each entry is ?UDELTN words long. The entire table (with eight entries) is ?UDLTH words long. (See Figure 2-79 for the structure of a map definition table entry and see Table 2-57 for a description of its contents.)

You must also set up a histogram packet of ?HWLTH words in your logical address space, and reserve a buffer to receive the histogram statistics. Load the packet address into AC2. (An offset within the packet points to the buffer address.) Figure 2-80 shows the structure of the histogram packet for 16-bit processes.

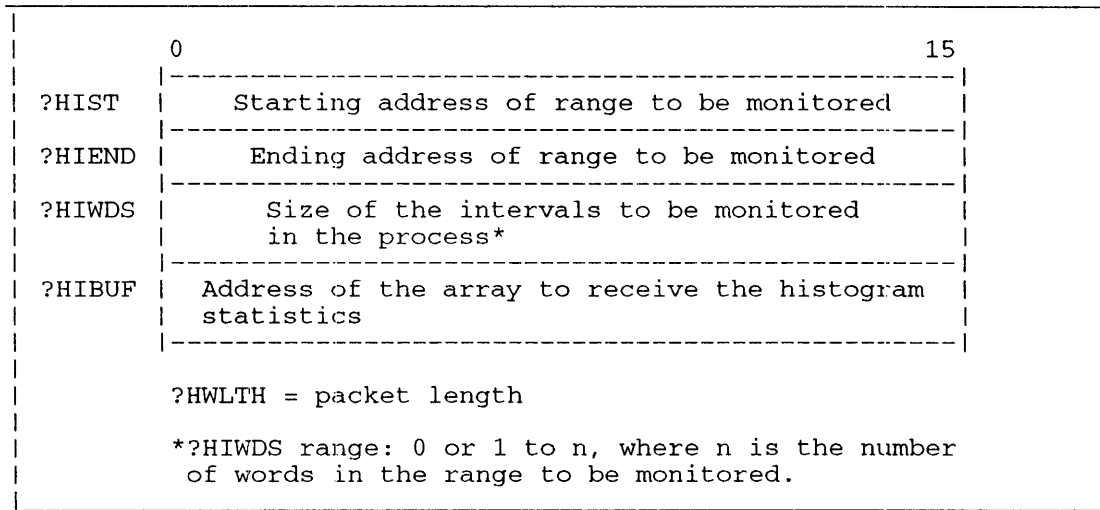


Figure 2-80. Structure of ?IHIST Packet

To start a simple histogram, which merely records how often the target process gained CPU control, set offset ?HIWDS to 0. This directs the operating system to ignore the contents of offsets ?HIST and ?HIEND and prevents the operating system from gathering range statistics. Set offset ?HIBUF to the address of the array you have reserved in your logical address space for the histogram statistics.

Each histogram array has two parts: a fixed-length header followed by double-precision array elements, which correspond to each interval that you want to monitor. Table 2-58 shows the structure of the histogram array for 16-bit processes.

Table 2-58. Structure of ?IHIST Array

Array Offset	Contents
?HTTH ?HTTL	Total number of real-time clock pulses (ticks) counted in this histogram.
?HPRH ?HPRL	Total number of ticks when the program counter (PC) was within the target process, but outside the specified range.
?HAPH ?HAPL	Total number of ticks in other processes.
?HSBH ?HSBL	Total number of ticks in the OS, except those recorded when it was in an idle loop.
?HSIH ?HSIL	Total number of ticks in a system idle loop.
?HARAY ?HARAY+1	Total number of ticks in the first interval.
...	
?HARAY+n*2-2	Total number of ticks in the nth interval.
?HARAY+n*2-1	

?ILKUP

Returns a global port number.

?ILKUP

error return

normal return

Input

AC0 Byte pointer to the pathname
of the target IPC entry

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Global port number
associated with the IPC
entry

AC2 File/device type of the IPC
file entry

Error Codes in AC0

ERFDE File does not exist

ERIFT Illegal file type (The file that you specified in AC1 is not an IPC file.)

ERVBP Invalid byte pointer passed as a system call argument

ER_FS_OBSOLETE_IPC_FILE_DETECTED

Obsolete IPC file type has been detected; file has been deleted (AOS/VS II only)

ER_FS_DIRECTORY_NOT_AVAILABLE

Directory not available because the LDU was force released (AOS/VS II only)

Why Use It?

?ILKUP returns the global port number of the IPC file that the target process previously created. Because a process must have the correct global port number to send an IPC message to another process, ?ILKUP can be a useful preliminary step to an ?ISEND or an ?IREC.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?ILKUP returns the global port number (which includes the PID, the ring number, and the local port number of the creator) associated with an IPC file entry that you specify. If the file does not exist or is not an IPC file, ?ILKUP fails and the operating system returns error code ERFDE or ERIFT in AC0.

Note that if you specify an IPC file whose local port number is 0, the number returned in AC1 is the port number of your terminal, not the port number of the file that you specified.

Notes

- See the descriptions of ?IREC and ?ISEND in this chapter.
- See PARU.32.SR or PARU.16.SR, for the IPC file and device types.

?IREC [*header address*]

error return

normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Address of the IPC header, unless you specify the address as an argument to ?IREC

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the IPC header

Error Codes in AC0

ERMPR	System call parameter address error (The receive buffer is not in the unshared area of your address space.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You would issue ?IREC either to receive an IPC message that another process sent via ?ISEND or to receive a process termination or connection management message.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?IREC opens a receiving port for the calling process, which allows the caller to receive an IPC message from a sending process. ?IREC requires a header, which lists, among other information, the origin and destination ports for the message.

Before you issue ?IREC, set up the header and reserve a receive buffer in your logical address space. You can cite the header address as an argument to ?IREC or load it into AC2 before you issue ?IREC. The ?IREC header consists of ?IPLTH words. Figure 2–81 shows the header's structure, and Table 2–60 describes its contents. Table 2–60 also describes the optional contents of ?ISFL in the ?IREC header.

Offset ?IOPH contains the global port number of the calling process (issue ?ILKUP to obtain this). If you set offset ?IOPH to 0, the calling process can accept messages from any sender. Similarly, if you set offset ?IDPN (the destination port number) to 0, the calling process can accept the message on any of its ports. During message transmission, the operating system writes the actual origin port number into offset ?IOPH, and the actual destination port number into offset ?IDPN.

Offset ?IPTR points to the receive buffer, which must be in the unshared area of your logical address space. (See Table 2–60 which describes the system flag words and user flag words.)

By default, the operating system suspends the receiver if there is no outstanding message for its ?IREC. You can avoid this by setting flag ?IFNBK in offset ?ISFL of the receive header. This flag signals the operating system to return an error to the receiver if there is no message.

?IREC Continued

An ?IREC from a particular ring can only receive an IPC message whose destination is that particular ring.

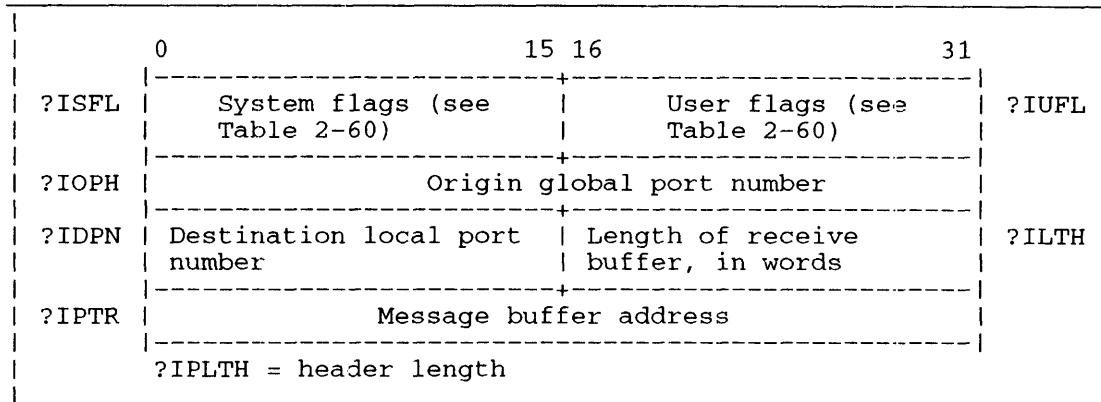


Figure 2-81. Structure of ?IREC Header

Table 2-60. Contents of ?IREC Header*

Offset	Contents
?ISFL	System flag word ?IFRFM--Receive a looped message (sent by this process to itself). Use this to test a process to see if both IPCs are working properly. Also, this is a good way for one program to communicate with another program in a single process (?CHAIN). ?IFSOV--Spool the message if the receive buffer is too small. ?IFNBK--Signal an error if there is no spooled message for this receiver. ?IFRING--Contains the sender's ring field (returned by the OS). (The ring field is a 3-bit field, Bits 13 through 15.) ?IFPR---Indicates .PR file type of sender; 0 if the sender is a 32-bit process, or 1 if the sender is a 16-bit process (returned by the OS).
?IUFL	User flag word. (See Figure 2-82.)
?IOPH (doubleword)	Origin global port number.
?IDPN	Destination local port number.
?ILTH	Size of the receive buffer, in words.
?IPTR (doubleword)	Address of the receive buffer.

* There is no default unless otherwise specified.

After ?IREC has finished you can identify the process that issued the associated ?ISEND or ?IS.R call. To do this, load the origin global port number from offset ?IOPH of the ?IREC packet into AC0 and then issue ?ISPLIT. After ?ISPLIT finishes AC1 will contain the PID of the sending process.

Sample Packet

```
HDR:   .BLK   ?IPLTH           ;Allocate enough space for the
                                     ;packet. Packet length = ?IPLTH.

       .LOC   HDR+?ISFL       ;System flags.
       .WORD  ?IFSOV         ;If buffer is too short, spool.

       .LOC   HDR+?IUFL       ;User flags.
       .WORD  0               ;Set to 0.

       .LOC   HDR+?IOPH       ;Global port number of calling
       .DWORD 0               ;process. (Set to 0 to accept
                                     ;messages from any sender.)

       .LOC   HDR+?IDPN       ;Destination port number.
       .WORD  0               ;Set to 0 to accept messages on any
                                     ;of the calling process's ports.

       .LOC   HDR+?ILTH       ;Receive buffer length (words).
       .WORD  80./2          ;Buffer length is 80./2. Reset
                                     ;buffer length after each ?IREC.

       .LOC   HDR+?IPTR       ;Receive buffer address (must be in
       .DWORD RECBUF        ;unshared area of your logical
                                     ;address space).

       .LOC   HDR+?IPLTH     ;End of ?IREC header.
```

Process Termination Messages

When a process terminates, the system sends a process termination IPC message to the father process (who created the terminating process). The system also sends connection management messages to the server (or customer) if the status of the connection in a customer/server relationship is affected. For a process to receive a process termination or connection management message, it must issue an ?IREC system call with offset ?IOPH set to a special system port, ?SPTM.

The location of the PID number, that ?IREC receives as part of a process termination or connection management message, depends on two things: the type of process that the receiver (i.e., the process that issues ?IREC) is, and the type of message that is received, as follows:

- A-type (smallPID) process, process termination message — the PID is in the right byte of offset ?IUFL. If ?ILTH contains 0, then the contents of ?IPTR are undefined. If ?ILTH doesn't contain 0, then ?IPTR contains a word pointer to a process termination message as described in the sections "Termination Messages for A-Type 32-bit Processes," and "Termination Messages for A-Type 16-bit Processes." For more information about process types see *Managing AOS/V S and AOS/V S II* (093-000541), and *AOS/V S System Concepts* (093-000335).
- B- or C-type (hybrid or anyPID programs) process, process termination message — the operating system returns 0 in the right byte of offset ?IUFL. If ?ILTH contains 0, then the contents of ?IPTR are undefined. If ?ILTH doesn't contain 0, then ?IPTR contains a word pointer to a process termination message as described in the section "Termination Messages for B-Type and C-Type Processes." For more information about process types see *Managing AOS/V S and AOS/V S II* (093-000541), and *AOS/V S System Concepts* (093-000335).
- A-type process, connection management (obituary) message — the PID is in the right byte of offset ?IUFL. Regardless of the contents of ?ILTH, the 24 high order bits of ?IPTR are undefined and the low order 8 bits contain a bit map according to the description of connection management in the *AOS/V S System Concepts* manual.

?IREC Continued

- B- or C-type process, connection management (obituary) message — the FID is in the 16 low order bits of offset ?IPTR (and the 16 high order bits are undefined). The contents of offset ?ILTH are unimportant, but the right byte of offset ?IUFL contains a bit map according to the description of connection management in the *AOS/VS System Concepts* manual.

An A-type process receives a termination message as defined in one of the following two places:

- Table 2-60 (page 2-296) and Table 2-63.1 (page 2-303)
- Table 2-60 (page 2-296) and Table 2-63 (page 2-301)

A B-type or C-type process receives a termination message whose receive buffers are defined in one of the following two places:

- Table 2-63.2 (page 2-304.2) in the section “Termination Messages for B-Type and C-Type Processes,” is for 32-bit processes.
- Table 2-64 (page 2-304.6) in the section “Termination Messages for B-Type and C-Type Processes,” is for 16-bit processes.

The operating system uses offset ?IUFL of the receive header to describe the reason for the Process Termination Message and to identify the process on whose behalf it is sending the message. Figure 2-82 shows the structure of offset ?IUFL.

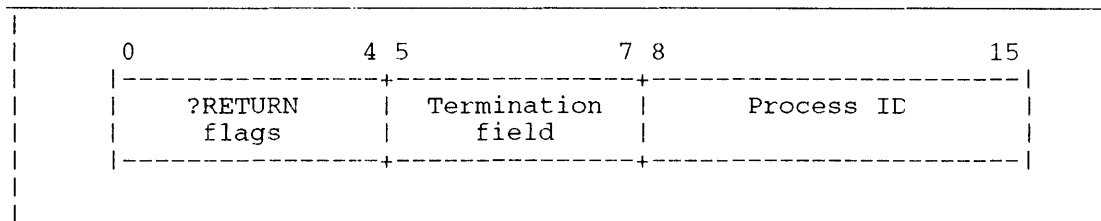


Figure 2-82. Structure of Offset ?IUFL

The termination field (bits 5 through 7) in offset ?IUFL is for the codes that the operating system uses to indicate why the termination message is being sent. The low order byte of offset ?IUFL always contains the PID of the process from which the message originated.

The termination field may contain any of the codes listed in Table 2-62, depending on the reason for the message. Note that when the termination field holds the value ?TEXT, you can find the real cause for the message in the first word of the message itself.

If a ?RETURN system call causes the termination message, the operating system uses the ?RETURN flags (bits 0 through 4) of the ?IUFL offset to provide additional information about the message format. See Table 2-61 for the ?RETURN flags.

If the message is in standard CLI format, the return flag is set to ?RFCF, and the remaining flags (?RFEC, ?RFA, ?RFE, and ?RFAB) have their conventional meanings. If the father is not the CLI, or if ?RFCF is not set, the father and son must agree about the contents of the message.

Table 2-61. Process Termination Codes in the ?RETURN Portion of ?IUFL

Code	Meaning
?RFCF	The termination message is in CLI format (the CLI is the father).
?RFEC	AC0 contains the error code.
?RFWA	A warning condition caused the termination.
?RFER	An error condition caused the termination.
?RFAB	An abort condition caused the termination.

If a Ctrl-C Ctrl-B (or Ctrl-C Ctrl-E) interrupt sequence caused the process to terminate, ?TCIN appears in the termination field of ?IUFL. If the operating system terminated the process because of an error condition, the system returns ?TAOS as the cause of termination. You can find a further explanation for the termination by looking at the error code returned in the ?IPTR offset of the message header.

If a customer/server relationship caused the message, the system returns codes ?TBCX, ?TCCX, or ?TABR in the termination field.

Termination Messages for A-Type 32-Bit Processes

When a 32-bit process terminates because of a ?TERM (without the optional message) or ?RETURN system call, the operating system sets the termination field of ?IUFL to ?TEXT and sets the first word of the termination message to ?T32T. If the message is due to a ?RETURN, the operating system uses the ?RETURN flags field of ?IUFL, and the message has the following format:

Word 0	?T32T (the extended termination code)
Word 1	Byte length of the ?RETURN message text
Words 2 and 3	Error code from ?RETURN AC0
Word 4	Start of message text

If the process terminated with a ?TERM with the optional message, the operating system forwards the message without modification.

If the 32-bit process terminated because of a user trap, the operating system sends a message in the format shown in Table 2-63. The length of this message is ?TPLN words.

Table 2-63. ?TEXT Code Termination Messages Sent on an A-Type 32-Bit Process User Trap

Word	Contents
0	?TR32 (the extended termination code).
1 and 2	AC0 contents.
3 and 4	AC1 contents.
5 and 6	AC2 contents.
7 and 8	AC3 contents.
9	Bit 0, carry; Bits 1 through 15, high-order bits of program counter.
10	Low-order bits of program counter.

(continued)

?IREC Continued

Table 2-63. ?TEXT Code Termination Messages Sent on an A-Type 32-Bit Process User Trap

Word	Contents
11	The following flag bits, which describe the trap
Bit 0=0	Trap occurred while control was in the user context.
Bit 0=1	Trap occurred while control was in the operating system.
Bit 3=1	A node time-out occurred. (This is a hardware error.)
Bit 4=1	Process tried to execute a privileged instruction.
Bit 5=1	Process tried to return to an inner ring from a subroutine call. (This is a violation of the ring structure.)
Bit 6=1	Process tried to issue a subroutine call to an outer ring. (This is a violation of the ring structure.)
Bit 7=1	Gate protection error. (This is a violation of the ring structure.)
Bit 8=1	Process tried to reference an address in an inner ring. (This is a violation of the ring structure.)
Bit 9=1	Process tried to read a read-protected page.
Bit 10=1	Process tried to execute data in an execute-protected area.
Bit 12=1	Process tried to write into a write-protected area.
Bit 13=1	Memory map validity error. (The process tried to refer to an address outside the user context.)
Bit 14=1	Defer error. (The process tried to use more than 16 levels of indirection in an address reference.)
Bit 15=1	Process tried to issue a machine-level I/O instruction without issuing the ?DEBL system call.

(concluded)

Termination Messages for A-Type 16-Bit Processes

When a 16-bit process terminates by issuing a ?RETURN or ?TERM system call, the system returns flag ?TSELF to the termination field in offset ?IUFL.

If the process terminated via the ?RETURN system call, the system sends one or more of the codes in Table 2-61 to the ?RETURN Flags field of ?IUFL. Furthermore, if ?RETURN was used, a 2-word header precedes the text of the message furnished by the ?RETURN:

- Word 0 Length of the ?RETURN message in bytes.
- Word 1 The error code (if any) found in ?RETURN's AC0.

The text of the message follows this header. If the process supplied no message, only the first two words appear.

If the process terminated itself via the ?TERM system call, the operating system returns either the termination message specified by the system call or returns nothing to the ?IREC receive buffer, if no message was sent.

If the 16-bit process terminated because of a user trap, the operating system sets the termination field to ?TRAP and sends the father a six-word message, as shown in Table 2-63.1.

If the process terminated because of an abort terminal interrupt (Ctrl-C Ctrl-B) or a ?TERM issued by a superior process, the operating system returns ?TCIN or ?TSUP, respectively, to the termination field, but sends no message.

Table 2-63.1. ?TRAP Termination Messages for A-Type 16-Bit Processes

Word	Contents
0	AC0 contents at the time of the trap.
1	AC1 contents at the time of the trap.
2	AC2 contents at the time of the trap.
3	AC3 contents at the time of the trap.
4	Bit 0, carry; Bits 1 through 15, program counter value.
5	The following flag bits, which describe the trap: <ul style="list-style-type: none"> Bit 0=0 Trap occurred while control was in the user context. Bit 0=1 Trap occurred while control was in the operating system. Bit 12=1 Process tried to write into a write-protected area. Bit 13=1 Memory map validity error. (The process tried to refer to an address outside the user context.) Bit 14=1 Defer error. (The process tried to use more than 16 levels of indirection in an address reference.) Bit 15=1 Process tried to issue a machine-level I/O instruction without issuing the ?DEBL system call.

?IREC Continued

Termination Messages for B-Type and C-Type Processes

B-type or C-type processes receive the new style termination message described on the next several pages. These formats apply only to issuing processes that are type B or type C.

The calling process is either a 32-bit one or a 16-bit one. Figure 2-82.1 describes the 32-bit termination message format, and Table 2-63.2 describes its contents. Figure 2-82.2 describes the 16-bit termination message format and Table 2-64 describes its contents.

Word ?IUFL of the termination IPC header contains the value of ?TEXT in the termination field, and the value indicates an extended termination message. Also, the PID field of ?IUFL contains 0.

If the ?TERM or ?RETURN call in the terminated process has a user-supplied message, then the values in words ?ISFL and ?IUFL of the user-supplied IPC header go to offsets ?TMR8 and ?TMR9 respectively.

A ?RETURN call has the return flags set in the IPC header for a user-supplied message. The default messages from a ?RETURN call are unchanged and will appear in their entirety in the ?TMMSG area of the termination message. Any user-supplied IPC to a ?TERM call will similarly be appended in this area. If the terminated process passes only an IPC header, then offset ?TMMLG contains zero and the first two words of offset ?TMMSG contain ?IPTR. Note that user-supplied messages are only appended to the default message; they never replace it.

	0	15 16	31	
?TMXTC	Extended termination code		Packet revision number	?TMPRV
?TMRS	Reserved (Set to 0.)		PID of terminated process	?TMPID
?TMUPD	Reserved (Set to 0.)			
	Reserved (Set to 0.)			
	Reserved (Set to 0.)			
	Reserved (Set to 0.)			
?TMR2	Reserved (Set to 0.)		Reserved (Set to 0.)	?TMR3
?TMAC0	AC0 of the terminated process			
?TMAC1	AC1 of the terminated process			
?TMAC2	AC2 of the terminated process			
?TMAC3	AC3 of the terminated process			
?TMCPC	C	Program Counter		
?TMSEC	Elapsed seconds since process creation			
?TMCPU	Milliseconds of CPU time used			
?TMLBK	Number of blocks read or written			
?TMPGS	Page-milliseconds used			
?TMPGD	Number of page faults since creation			
?TMPGF	Number of page faults -- no disk I/O			
?TMR4	Reserved (Set to 0.)		Reserved (Set to 0.)	?TMR5
?TMR6	Reserved (Set to 0.)		Reserved (Set to 0.)	?TMR7
?TMR8	Reserved (Set to 0.)		Reserved (Set to 0.)	?TMR9
?TMTCD	Trap code		Message length	?TMMLG
?TMMSG	First word of default message		Second word of default message	?TMMSG + 1
?TMMSG + 2	First word of user supplied message		Second word of user supplied message	?TMMSG + 3
.	.			.
.	.			.
?TMMSG + (n-2)	Next to last word of user supplied msg.		Last word of user supplied message	?TMMSG + (n-1)
	?TDFL = packet length for no user supplied message			

Figure 2-82.1. Structure of Termination Message from a 32-bit B- or C-Type Process

?IREC Continued

Table 2-63.2. Contents of Termination Message from a 32-bit B- or C-Type Process

Offset	Contents
?TMXTC	The operating system returns the extended termination code to indicate the type of process termination. The values and meanings of these codes are as follows. ?XT16T -- A 16-bit termination of self occurred. ?XTR16 -- A 16-bit user trap occurred. ?XTCIN -- Termination by a terminal interrupt. ?XTSUP -- Termination by a superior process. ?XTAOS -- Termination by AOS. ?XTBCX -- Customer connection broken. ?XTCCX -- Customer chained. ?XTABR -- Customer did ?TABT. ?XT32T -- A 32-bit termination of self occurred. ?XTR32 -- A 32-bit user trap occurred.
?TMPRV	Packet revision number. Place ?TM6 here.
?TMRS	Reserved. Set to 0.
?TMPID	The operating system returns the PID of the terminated process.
?TMUPD (8. words)	Reserved. Set to 0.
?TMR2	Reserved. Set to 0.
?TMR3	Reserved. Set to 0.
?TMAC0 (double- word)	The operating system returns the contents of AC0 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). The high-order word is undefined if a 16-bit process terminated.
?TMAC1 (double- word)	The operating system returns the contents of AC1 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). The high-order word is undefined if a 16-bit process terminated.
?TMAC2 (double- word)	The operating system returns the contents of AC2 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). The high-order word is undefined if a 16-bit process terminated.
?TMAC3 (double- word)	The operating system returns the contents of AC3 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). The high-order word is undefined if a 16-bit process terminated.

(continued)

Table 2-63.2. Contents of Termination Message from a 32-bit B- or C-Type Process

Offset	Contents
?TMCP (double- word)	The operating system returns the carry bit and program counter contents at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). The high-order word is undefined if a 16-bit process terminated.
?TMSEC (double- word)	The operating system returns the number of elapsed seconds since the process was created.
?TMCPU (double- word)	The operating system returns the number of milliseconds of CPU time the process used.
?TMBLK (double- word)	The operating system returns the number of blocks read or written.
?TMPGS (double- word)	The operating system returns the page usage over CPU time in pages/second.
?TMPGD (double- word)	The operating system returns the number of page faults since the process was created.
?TMPGF (double- word)	The operating system returns the number of page faults, with no disk I/O, since the process was created.
?TMR4	Reserved. (Set to 0.)
?TMR5	Reserved. (Set to 0.)
?TMR6	Reserved. (Set to 0.)
?TMR7	Reserved. (Set to 0.)
?TMR8	Reserved. (Set to 0.)
?TMR9	Reserved. (Set to 0.)
?TMTCD	The operating system returns the trap code.
?TMMLG	The operating system returns the number of words in any user-supplied message. This number might be zero.
?TMMSG	Beginning of the message area, starting with two words; ending with the zero or more words in any user-supplied message.

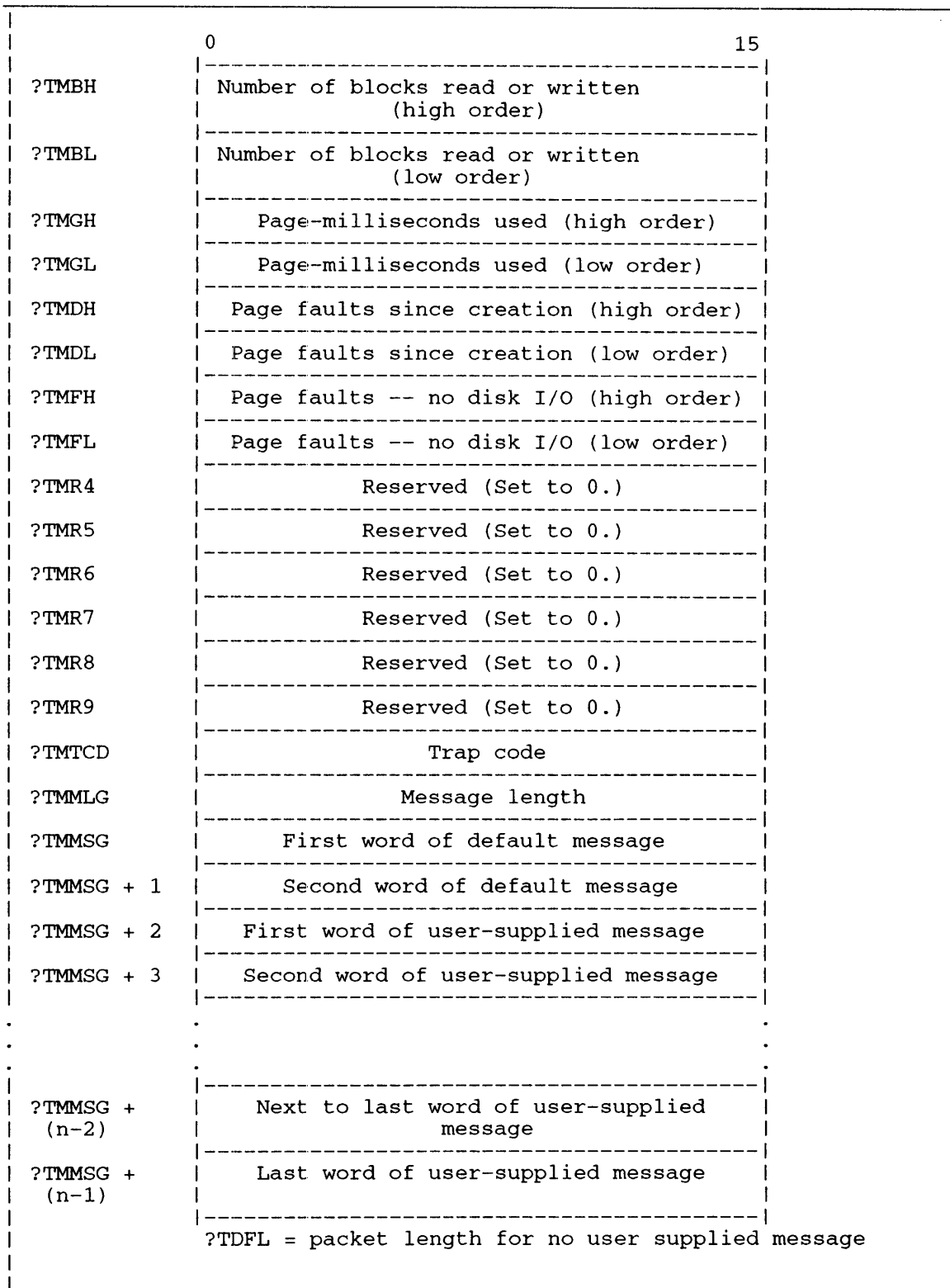
(concluded)

?IREC Continued

	0	15
?TMXTC	Extended termination code	
?TMPRV	Packet revision number	
?TMRS	Reserved (Set to 0.)	
?TMPID	PID of terminated process	
?TMUPD	Reserved (Set to 0.)	
?TMUPD + 1	Reserved (Set to 0.)	
.	.	.
?TMUPD + 7	Reserved (Set to 0.)	
?TMR2	Reserved (Set to 0.)	
?TMR3	Reserved (Set to 0.)	
?TMOH	AC0 of terminated process (high order)	
?TMO L	AC0 of terminated process (low order)	
?TM1H	AC1 of terminated process (high order)	
?TM1L	AC1 of terminated process (low order)	
?TM2H	AC2 of terminated process (high order)	
?TM2L	AC2 of terminated process (low order)	
?TM3H	AC3 of terminated process (high order)	
?TM3L	AC3 of terminated process (low order)	
?TMLH	C Program Counter (high order)	
?TMLL	C Program Counter (low order)	
?TMSH	Elapsed seconds since process creation (high order)	
?TMSL	Elapsed seconds since process creation (low order)	
?TMCH	Milliseconds of CPU time used (high order)	
?TMCL	Milliseconds of CPU time used (low order)	

(continued)

Figure 2-82.2. Structure of Termination Message from a 16-bit B- or C-Type Process



(concluded)

Figure 2-82.2. Structure of Termination Message from a 16-bit B- or C-Type Process

?IREC Continued

Table 2-64. Contents of Termination Message from a 16-bit B- or C-Type Process

Offset	Contents
?TMXTC	The operating system returns the extended termination code to indicate the type of process termination. The values and meanings of these codes are as follows. ?XT16T -- a 16-bit termination of self occurred. ?XTR16 -- a 16-bit user trap occurred. ?XTCIN -- termination by a terminal interrupt. ?XTSUP -- termination by a superior process. ?XTAOS -- termination by AOS. ?XTBCX -- customer connection broken. ?XTCCX -- customer chained. ?XTABR -- customer did ?TABT. ?XT32T -- a 32-bit termination of self occurred. ?XTR32 -- a 32-bit user trap occurred.
?TMPRV	Packet revision number. Place ?TM6 here.
?TMRS	Reserved. (Set to 0.)
?TMPID	The operating system returns the PID of the terminated process.
?TMUPD (8 words)	Reserved. (Set to 0.)
?TMR2	Reserved. (Set to 0.)
?TMR3	Reserved. (Set to 0.)
?TM0H	The operating system returns the high-order contents of AC0 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). This offset is undefined if a 16-bit process terminated.
?TM0L	The operating system returns the low-order contents of AC0 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32).
?TM1H	The operating system returns the high-order contents of AC1 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). This offset is undefined if a 16-bit process terminated.
?TM1L	The operating system returns the low-order contents of AC1 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32).
?TM2H	The operating system returns the high-order contents of AC2 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). This offset is undefined if a 16-bit process terminated.
?TM2L	The operating system returns the low-order contents of AC2 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32).

(continued)

Table 2-64. Contents of Termination Message from a 16-bit B- or C-Type Process

Offset	Contents
?TM3H	The operating system returns the high-order contents of AC3 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). This offset is undefined if a 16-bit process terminated.
?TM3L	The operating system returns the low-order contents of AC3 at the time of the termination. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32).
?TMLH	The operating system returns the carry bit and high-order program counter contents if and when a 32-bit process terminated. This offset is valid only on a trap (?TMXTC contains ?XTR16 or ?XTR32). The offset is undefined if a 16-bit process terminated.
?TMLL	The operating system returns the low-order program counter contents if and when a 32-bit process terminated, or else the carry bit and program counter contents if and when a 16-bit process terminated. The offset is valid only on a trap (?TMXTC contains ?XTR15 or ?XTR32).
?TMSH	The operating system returns the number of elapsed seconds since the process was created (high order).
?TMSL	The operating system returns the number of elapsed seconds since the process was created (low order).
?TMCH	The operating system returns the number of milliseconds of CPU time the process used (high order).
?TMCL	The operating system returns the number of milliseconds of CPU time the process used (low order).
?TMBH	The operating system returns the number of blocks read or written (high order).
?TMBL	The operating system returns the number of blocks read or written (low order).
?TMGH	The operating system returns the page usage over CPU time in pages/second (high order).
?TMGL	The operating system returns the page usage over CPU time in pages/second (low order).
?TMDH	The operating system returns the number of page faults since the process was created (high order).
?TMDL	The operating system returns the number of page faults since the process was created (low order).

(continued)

?IREC Continued

Table 2-64. Contents of Termination Message from a 16-bit B- or C-Type Process

Offset	Contents
?TMFH	The operating system returns the number of page faults, with no disk I/O, since the process was created (high order).
?TMFL	The operating system returns the number of page faults, with no disk I/O, since the process was created (low order).
?TMR4	Reserved. (Set to 0.)
?TMR5	Reserved. (Set to 0.)
?TMR6	Reserved. (Set to 0.)
?TMR7	Reserved. (Set to 0.)
?TMR8	Reserved. (Set to 0.)
?TMR9	Reserved. (Set to 0.)
?TMTCD	The operating system returns the trap code.
?TMLLG	The operating system returns the number of words in any user-supplied message. This number might be zero.
?TMSG	Beginning of the message area, starting with two words; ending with the zero or more words in any user-supplied message.

(concluded)

Notes

- See the descriptions of ?ILKUP, ?TERM, ?RETURN, and ?ISEND in this chapter.

?IRMV

Removes a user device.

?IRMV

error return

normal return

Input

AC0 Device code of the user device that you want to remove

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Undefined

AC2 Undefined

Error Codes in AC0

ERDNM Illegal device code
ERPRE Invalid system call parameter
ERPRV Caller not privileged for this action
ERPTY Illegal process type

Why Use It?

?IRMV revokes a previous ?IDEF; that is, it allows you to remove a device that you defined earlier in your program. ?IRMV removes only a specific user-defined device.

Who Can Use It?

To issue ?IRMV, a process must have privilege ?PVDV. There are no restrictions concerning file access.

What It Does

?IRMV removes a device's DCT entry from the interrupt vector table. After the operating system executes ?IRMV, it ignores the target device and all subsequent interrupts from it.

Before you issue ?IRMV, load AC0 with the device code you specified when you defined the device with ?IDEF.

Notes

- See the description of ?IDEF in this chapter.

Sample Header

```
HDR:  .BLK    ?IPLTH      ;Header length.
      .LOC    HDR+?ISFL  ;System flags.
      .WORD   0          ;Set to 0.
      .LOC    HDR+?IUFL  ;User flags.
      .WORD   0          ;Set to 0.
      .LOC    HDR+?IDPH  ;Global port number of receiving
      .DWORD  5          ;process. (?ILKUP and ?TPORT return
                        ;this information.)
      .LOC    HDR+?IOPN  ;Origin port no.
      .WORD   7          ;Local port no. to use for this
                        ;?ISEND.
      .LOC    HDR+?ILTH  ;Message buffer length (words).
      .WORD   7          ;Set buffer length before each
                        ;?ISEND.
      .LOC    HDR+?IPTR  ;Message buffer address (must be in
      .DWORD  SBUFF     ;unshared area of your logical
                        ;address space).
      .LOC    HDR+?IPLTH ;End of ?ISEND header.
```

Notes

- See the descriptions of ?IREC, ?IS.R, ?ILKUP, and ?TPORT in this chapter.
- In the explanation of ?PROC, see the section “Offset ?PIPC.” It explains how ?PROC can send a CLI-format command line as the IPC message.
- A global port number contains the PID of the target process. In order to ensure the global port number remains unique, a connection must exist between the calling and the target processes. This connection will cause the operating system to reserve the PID portion of the global port until the connection is explicitly broken. If an ?ISEND is attempted when the process described by the global port number has terminated, connection management will guarantee the ERIDP error will be returned until the calling process does the ?DCON. See the description of ?CON, ?DCON, ?DRCON, ?RESIGN?, and ?SERVE in this chapter.

?ISPLIT

Finds the owner of a port (including its ring number).

?ISPLIT

error return

normal return

Input

- AC0 One of the following:
- Global port number (32-bit users only)
 - OPH, which indicates high-order bits of global port number (16-bit users only)
- AC1 One of the following:
- Reserved (Set to 0.) (32-bit users only)
 - OPL, which indicates low-order bits of global port number (16-bit users only)
- AC2 Reserved (Set to 0.)

Output

- AC0 One of the following:
- Ring field in Bits 29 through 31 (32-bit users only)
 - Ring field in Bits 13 through 15 (16-bit users only)
- AC1 PID of the process that owns the port
- AC2 Local port number

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?ISPLIT, like ?IMERGE, allows both 16- and 32-bit users to manipulate ring fields within global port numbers.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?ISPLIT returns the ring field, the PID, and the local port number associated with the global port number you specify in AC0.

Notes

- See the description of ?IMERGE in this chapter.

Offset ?LBFG contains status bits that allow you to specify the tape unit's density mode and whether you want the label in IBM format. There are three density mode settings (comparable to those in the ?OPEN packet): ?LB8 sets the tape unit to a density of 800 bytes/inch; ?LB16 sets the unit to 1600 bytes/inch; ?LB62 sets the tape unit to a density of 6,250 bytes/inch; ?LBAM directs the operating system to set the correct density automatically.

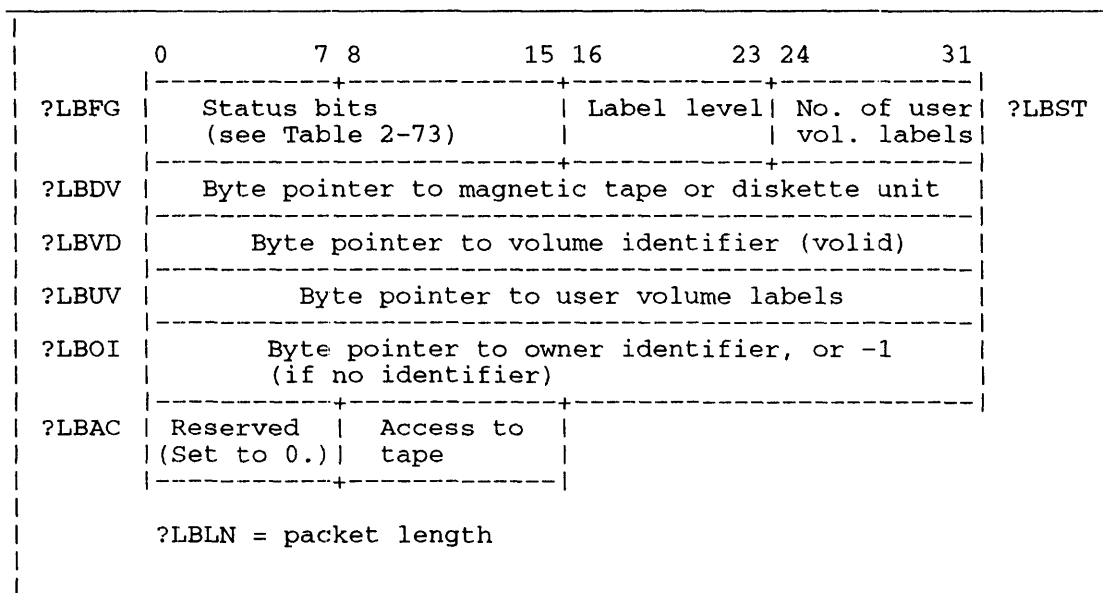


Figure 2-91. Structure of ?LABEL Packet

If you default the density mode, the operating system sets the unit to the density specified in the system-generation procedure. Before you default this parameter, make sure the tape's density matches the density specification set during the system-generation procedure.

Parameter ?LBSC in offset ?LBFG ("scratch this tape") causes the operating system to "scratch" (overwrite) all data on the tape or diskette as it supplies the label.

Use the right byte of offset ?LBAC to specify access to the media. Typically, you supply a space character (ASCII 040) in this byte, which gives all users full access to the media.

?LABEL Continued

Table 2-73. Contents of ?LABEL Packet*

Offset	Contents
?LBFG	Status bits ?LBIM--Label is in IBM format (tape only). ?LBSC--Scratch this tape/diskette. ?LBMF--Use buffered mode (MTJ Model 6352 tape only). ?LBMS--Use streaming mode (MTJ Model 6352 tape only). Density mode (tape only): ?LB8--800 bytes/inch. ?LB16--1600 bytes/inch. ?LB62--6250 bytes/inch. ?LBAM--automatic density matching. ?LB5--Low tape density. ?LB6--Medium tape density. ?LB7--High tape density. Error handling (diskette only): ?LBMR--Check for bad blocks and remap if necessary. ?LBMP--Check for bad blocks and return ERBDK if any are present.
?LBST	Left byte: label level. Supply 1, 2, 3, or 4; the default is 3. Right byte: number of user volume labels (tape only). Supply 1, 2, 3, ..., or 9.
?LBDV (doubleword)	Byte pointer to magnetic tape or diskette unit.
?LBVD (doubleword)	Byte pointer to volume identifier (valid).
?LBUV (doubleword)	Byte pointer to user volume labels (tape only). You can have a maximum of 76 bytes in these labels and you must separate the labels with the null character.
?LBOI (doubleword)	Byte pointer to owner identifier or -1 (if no identifier).
?LBAC	Left byte: reserved. (Set to 0.) Right byte: accessibility (access to tape).

*There is no default unless otherwise specified.

?OPEN [*packet address*]

error return

normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Address of the ?OPEN packet, unless you specify the address as an argument to ?OPEN

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?OPEN packet

Error Codes in AC0

EREO1	File is open, can't exclusively open
EREO2	File exclusively opened, can't open
ERFAD	File access denied
ERIFL	IAC (Intelligent Asynchronous Controller) failure
ERINP	IPC file not opened by another proc
ERIPS	Illegal pipe size
ERMPR	System call parameter address error
ERP00	Illegal pipe open option
ERVWP	Invalid word pointer passed as a system call argument

Who Can Use It?

There are no special process privileges needed to issue this call. You need Execute access to the file's directory. You also need Write access to the file's directory if issuing ?OPEN will create the file. And, you need Read access to the file and its directory if you are issuing ?OPEN against a file that presently exists.

What It Does

?OPEN opens a file or device for I/O and directs the operating system to assign it a unique channel number. A single process can open as many as 224 channels at the same time. Note that you cannot use ?OPEN to open a file for block I/O, physical block I/O, shared access, or to open a binary synchronous communications line.

You can use ?OPEN to create and then open a file. If you choose this creation option and you choose the default file-type specification, the operating system creates the file as a user data file. (User data files are not executable program files.)

The packets for ?OPEN, ?READ, ?WRITE, and ?CLOSE have the same structure, although not all offsets apply to every system call. You can specify the packet address as an argument to ?OPEN or load AC2 with the packet address before you issue ?OPEN. In both cases, the operating system returns the packet address in AC2.

?OPEN Continued

Figure 2–113 shows the structure of the ?OPEN packet, and Table 2–90 describes each offset and bit position in the packet. The accompanying text explains the options you can exercise in setting the packet specifications.

Figure 2–113 includes byte pointers to extensions for the basic I/O packet. These extensions direct the operating system to perform extended processing on the target file. See the following section “Packet Versions.”

The following guidelines apply to the ?OPEN packet:

- You can default some specifications, such as record format, and use the values you set for them when you created the file. You can also alter some of the ?OPEN specifications, such as record format and buffer address, when you read to or write from the file.
- When you close and reopen a file, the operating system overwrites the default values in the packet. Thus, you must reset these values before you use the packet again.
- The parameters in the file specifications word (?ISTI) represent bit masks, not individual bits. To select more than one option for ?ISTI, OR the appropriate masks. For example, the ?ISTI specification ?IEXO!?OFIN opens the file for the exclusive use of the calling task for read purposes only.
- You must set all unused bits in offset ?ISTI to 0.
- The operating system returns the file’s channel number to offset ?ICH in the I/O packet. (The operating system always sets the channel number, even if you place a value in ?ICH.)
- You need not set the offsets used by ?READ and ?WRITE to 0, because the operating system ignores them when it executes ?OPEN.

Packet Versions

There are three versions of the ?OPEN parameter packet as follows. Refer to Figure 2–113 as you read the following summaries of these three versions.

Standard	which includes offsets ?ICH through ?IDEL, and is ?IOSZ words long.
Short extended	which includes offsets ?ICH through ?ENET, and is ?IBLT words long. Set bit ?IPKL in offset ?ISTI to indicate the short extended packet.
Long extended	which includes offsets ?ICH through ?ENET and 24 reserved words for a length of ?ETMX words. Set bit ?IPKL in offset ?ISTI and bit ?IMP2 in offset ?ISTO to indicate the long extended packet. You should set the 24 reserved words to zero with one exception. The first doubleword, offset ?ETER, contains zero. The second doubleword, offset ?ETSN, contains 0. The third doubleword (offset ?EPIP, the exception) contains 1S0 and a word pointer to the optional pipe extension packet. (?OPEN uses the third doubleword when working with a pipe file.)

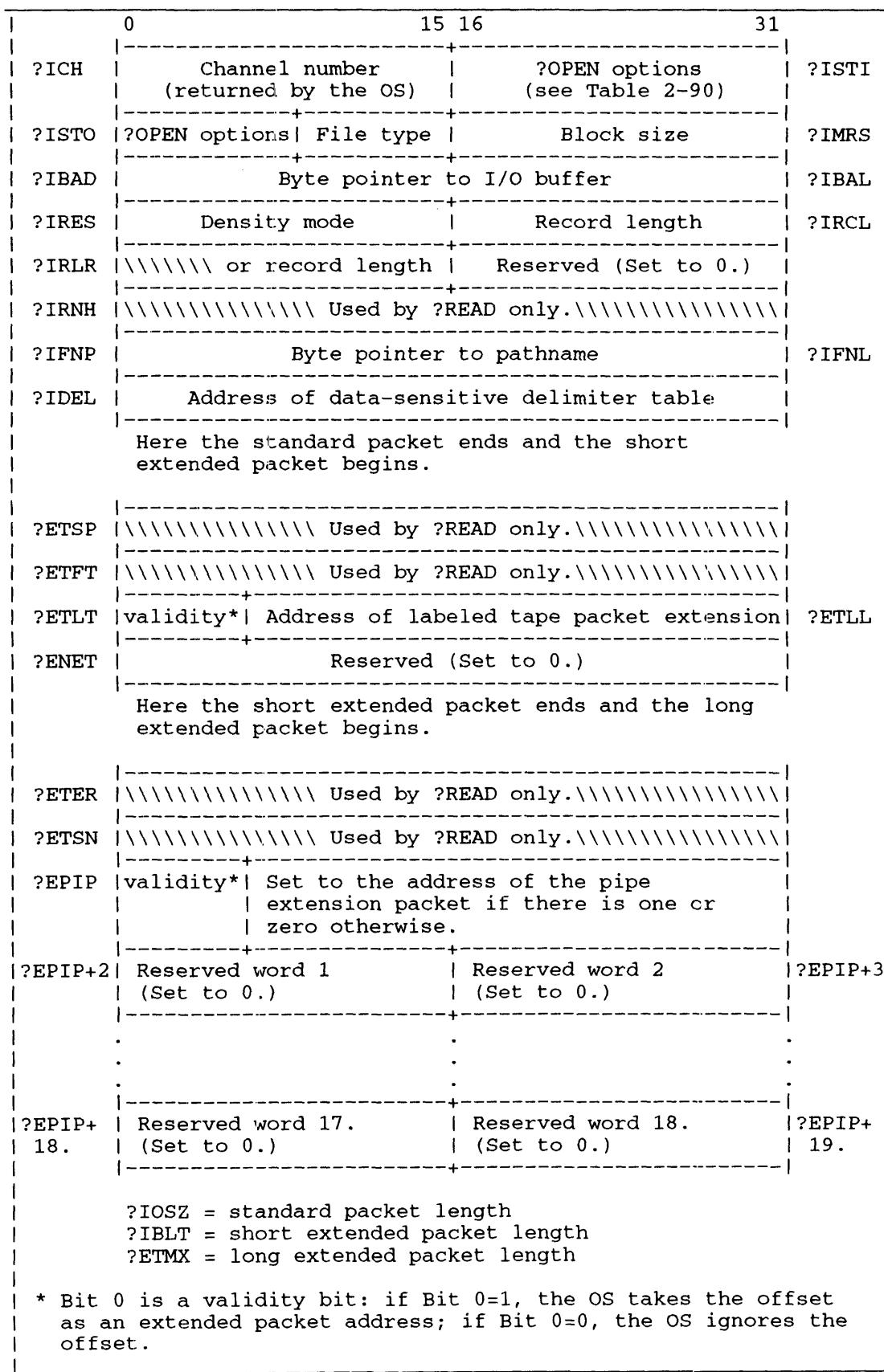


Figure 2-113. Structure of ?OPEN Packet

?OPEN Continued

Table 2-90. Contents of ?OPEN Packet*

Offset	Contents
?ICH	Channel number (assigned by the OS).
?ISTI	File specifications word: Packet type ?IPKL--Extended packet. DEFAULT = 0 (no extended packet) Format select ?ICRF--Change format to that specified in record format field. DEFAULT = 0 (use record format specified in the ?CREATE packet). Absolute file pointer position (?READ/?WRITE only). Append ?APND--Open the file for appending. DEFAULT = 0 (set file pointer to first word in file). Binary I/O. Normally, only ?READ/?WRITE supplies a value here. However, if you issue ?OPEN against a queue type file (e.g., @LPT) and specify ?IBIN here, AOS/VS responds to this bit. Specifi- cally, AOS/VS would do a binary queue submission to EXEC. Force output option (?READ/?WRITE only). Exclusive open ?IEXO--Permit no other task in this or any other process to open file until it has been closed. DEFAULT = 0 (nonexclusive open). Pipe files require a nonexclusive open.

* There is no default unless otherwise specified. (continued)

Table 2-90. Contents of ?OPEN Packet*

Offset	Contents
===== ?ISTI (continued)	<p>Priority read</p> <p>?PDEL--Open file for priority read.</p> <p>DEFAULT = 0 (file has normal priority).</p> <p>Creation option</p> <p>?OFCE--If the file exists, open the file. If the file does not exist, create one and open it.</p> <p>?OFCR--If the file exists, return an error. If the file does not exist, create one and open it.</p> <p>?OFCR! ?OFCE--If the file exists, then delete it, recreate it, and open it. If the file does not exist, then create it and open it.</p> <p>DEFAULT = 0 (if file does not exist, do not create it; return ?OPEN error).</p> <p>Input/output</p> <p>?OFIN--Open for input. ?OFOT--Open for output. ?OFIO--Open for input/output. (Not available for pipe files.)</p> <p>Record format</p> <p>?RTDY--Dynamic-length. ?RTDS--Data-sensitive. ?RTFX--Fixed-length. ?RTVR--Variable-length. ?RTUN--Undefined-length. ?RTVB--Variable block, variable record.</p> <p>IPC no wait option</p> <p>?IIPC--Setting "IPC No Wait" on IPC file</p> <p>When you select ?IIPC, the process that initially opens the IPC file will not wait for the synchronization message from the partner process. Normally the process waits for the partner process to open the IPC file.</p>
?ISTO	<p>Left byte</p> <p>?SHOP--Shared ?OPEN request. (Not available for pipe files.)</p> <p>?IMFF--Inhibit initial form feed. ?IMP2--Long extended packet. ?IMHN--Hold nonpriority reads. DEFAULT = 0 (normal open; write initial form feed when opening line printers).</p> <p>If you specify none of the options ?SHOP through ?IMHN, you must specify 0 here.</p>

* There is no default unless otherwise specified. (continued)

?OPEN Continued

Table 2-90. Contents of ?OPEN Packet*

Offset	Contents
?ISTO	<p>Right byte File Type. (See Table 2-92)</p> <p>DEFAULT = 0 (if file exists, the OS ignores this parameter and uses the file type specified in the ?CREATE packet). If you are creating the file (you set ?OFCR and/or ?OFCE in ?ISTI), the file type is ?FUDF (user data file).</p>
?IMRS	<p>Physical block size (in Kbytes). Specify the length of the pipe in 2Kbyte (page) multiples, up to a 20 page maximum. 1 page is added to the pipe length at creation.</p> <p>DEFAULT = -1 (block size = 2 Kbytes)</p>
?IBAD (doubleword)	<p>Byte pointer to record I/O buffer.</p> <p>DEFAULT = -1 (deferred until you issue ?READ or ?WRITE).</p>
?IRES	<p>Density mode (for magnetic tapes only). Set this field to 0 for all other file and device types.</p> <p>?IDAM--Automatic density matching. ?ID8--Density 800 bytes/inch. ?ID16--Density 1600 bytes/inch. ?ID62--Density 6250 bytes/inch. ?ID5--Low tape density. ?ID6--Medium tape density. ?ID7--High tape density.</p> <p>DEFAULT = 0 (use density mode specified during the system-generation procedure).</p> <p>Transfer mode (for Model 6352 magnetic tape units only)</p> <p>?OMBFM--Use buffered mode when performing tape I/O. ?OMSTR--Use streaming mode when performing tape I/O. ?IMNTE--Use the emulation override. ?IMCOF--Data compression off. ?OIBM--Open as an IBM-labelled tape. ?OANS--Open as an ANSI-labelled tape.</p> <p>DEFAULT = 0 (open as an AOS/VS or AOS/RT32 tape).</p>

* There is no default unless otherwise specified.

(continued)

The ?SHOP parameter causes ?OPEN to behave much like ?SOPEN. However, unlike ?SOPEN, ?OPEN with ?SHOP set involves the Agent. Therefore, to minimize system overhead, you should use ?SOPEN. Using ?SHOP shared I/O consumes more of the system I/O resources (than unshared I/O) and you may not be able to open as many channels.

Normally, the operating system generates a form feed when it opens a line printer. If you do not want this initial form feed, set bit ?IBFF in offset ?ISTO.

Other ?OPEN Offsets

Offset ?IBAD points to the I/O buffer. The I/O buffer is an area that you set up to receive the records you will later read or write. The I/O buffer must be large enough to accommodate the largest record you will read or write; if it is not, the operating system returns error code ERMPR when it tries to perform the I/O system call. If you do not specify an I/O buffer when you open the file, you must do so every time you issue ?READ or ?WRITE.

The meaning of offset ?IRCL varies, depending on the file's record format. If the file consists of fixed-length records, set ?IRCL to the record length in bytes. If the file consists of dynamic-length, variable-length or data-sensitive records, set ?IRCL to the number of bytes to be transferred (read or written) upon each I/O request. If the file consists of fixed-length records and ?IRCL contains -1, ?OPEN places the length of each record in the file in offset ?IRLR.

If the record type is data-sensitive and the operating system encounters a delimiter before it reaches the specified number of bytes, it terminates the I/O transfer. However, if the record type is data-sensitive and the operating system does not find a delimiter within ?IRCL bytes, it returns error code ERLTL ("line too long").

Setting a Delimiter Table

To define alternative delimiters for data-sensitive records, set up a delimiter table in your logical address space, and specify its address in the ?OPEN offset ?IDEL.

The delimiter table must consist of 16 consecutive 16-bit words, to form a table of 256 bits. Each bit in the table represents an ASCII character. Reading from left to right, the first bit (Word 0, Bit 0) represents the null character (0), the second bit (Word 0, Bit 1) represents the Ctrl-A character (octal 001), and so forth. For each bit you set, the operating system recognizes the corresponding character as a delimiter for the file's records.

Figure 2-114 depicts a sample delimiter table with bits set to make the null, carriage return (015 octal), and rubout (177 octal) characters data-sensitive delimiters.

?OPEN Continued

	Bit 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Word 0	*													**		
Word 1																
Word 2																
....																
Word 7																++
....																

Key: * null (000)
 ** carriage return (015 octal)
 ++ rubout (177 octal)

Figure 2-114. Sample Delimiter Table

You can also use the ?SDLM system call to set a delimiter table for an open record or open device. If you issue ?SDLM after you issue ?OPEN, the ?SDLM delimiter specifications override those in the ?OPEN packet.

Extension Packet for Pipes

You use one extension packet for pipes, and only ?OPEN provides the pipe extension packet. You supply the packet as an extension to the packet for a ?OPEN, ?READ, or ?WRITE system call. You must specify the pipe size in 2 Kbyte multiples in offset ?IMRS (see Table 2-90). The maximum pipe size is 20 pages (2 Kbytes equal one page), or you can use the default value of -1.

When the pipe file is created, the operating system adds a 1-page overhead to the size specified in ?IMRS. For a full description of using pipes, see *AOS/VS System Concepts* (093-000335).

Figure 2-115 contains the extension packet. Table 2-93 describes its contents.

	0	15	16	31	
?PIRV	Packet revision		Reserved (Set to 0.)		?PIRS
?PIFG	Flags				
?PIPD	Pending behavior code		Reserved (Set to 0.)		?PITI
?PIPI	Reserved (Set to 0.)				
	?PILN = extension packet length				

Figure 2-115. Structure of ?OPEN Extension Packet for Pipes

Table 2-93. Contents of ?OPEN Extension Packet for Pipes

Offset	Contents
?PIRV	Packet revision number. Place the value of symbol ?PKR0 here.
?PIRS	Reserved word. (Set to 0.)
?PIFG	Flag word. (Set to 0.)
?PIPD	Use the following symbols to specify the operating system's action. <ul style="list-style-type: none"> ?PALW -- Always pend on a read from an empty pipe or on a write to a full pipe, regardless of whether the pipe is one-ended or two-ended. ?PTWO -- Pend on a read from an empty pipe or on a write to a full pipe, but only if the pipe is two-ended. If the pipe is one-ended, then return error code EREOF on a read or else ERPFL on a write. ?PTWO is the default value of this offset. ?PNVR -- Never pend on a read from an empty pipe or on a write to a full pipe, regardless of whether the pipe is one-ended or two-ended. Return error code EREOF on a read or else ERPFL on a write.
?PITI	Reserved word. (Set to 0.)
?PIPI (double- word)	Reserved word. (Set to 0.)

Magnetic Tapes

Offset ?IMRS specifies the physical block size of a magnetic tape or disk file. The physical block size is the number of bytes, minus 1, in the largest physical block. The operating system uses this parameter to allocate a buffer ?IMPSTL bytes long for I/O. (The default buffer size is 2 Kbytes (4 blocks) for disk files.)

If you are opening a magnetic tape on a controller that supports multiple tape densities (such as a type MTB controller), you can set offset ?IRES to a specific density mode for the tape. There are seven density mode settings:

- ?IDAM directs the operating system to set the tape unit to the correct density mode automatically.
- ?ID8 sets the tape unit to a density mode of 800 bytes/inch.
- ?ID16 sets the tape unit to a density mode of 1600 bytes/inch.
- ?ID62 sets the tape unit to a density mode of 6250 bytes/inch.
- ?ID5 sets the tape unit to the lowest density it supports.
- ?ID6 sets the tape unit to the middle density on drives supporting three densities, and to the lower density on drives supporting two densities.
- ?ID7 sets the tape unit to the highest density it supports.

?OPEN Continued

If you set offset ?IRES to 0 (the default), the operating system uses the density mode that was set during the system-generation procedure. However, if you set offset ?IRES to 0, you must make sure that the tape's density mode matches the default mode. Otherwise, ?OPEN fails after I/O is attempted.

There are two other possible density mode errors: ERCND (controller does not support this density mode), and ERITD (indecipherable tape density). The operating system returns ERITD when either the tape or the tape unit is damaged.

Labeled Magnetic Tapes

If you want to read or write additional user labels on a labeled tape file, you must open the file with the ?OPEN packet extension for labeled tapes. To do this, perform the following steps:

1. Set Bit ?IPKL in offset ?ISTI of the standard ?OPEN packet.
2. Set Bit 0 of offset ?ETLT to 1 (to indicate that there is a packet extension).

The first time you use the labeled tape extension, you must set Bit 0 of ?ETLT (the flag bit) to 1 so that the system will recognize Bits 1 through 31 of offset ?ETLT as the address of the packet extension. The operating system sets the flag bit to 0 after it executes the ?OPEN.

If you want to change one or more of the specifications, and then reuse the extension, be sure to reset the flag bit to 1. If the flag bit is 0, the operating system assumes that the packet extension has not changed and, therefore, does not re-examine it.

You can set aside space for the labeled tape extension even if you do not supply values for it. In this case, the operating system simply fills the packet extension with the existing label information.

3. Set Bits 1 through 31 of offset ?ETLT to the address of the packet extension.
4. Precede the labeled tape extension with the following pseudo-op to reserve the correct number of words for the packet extension:

```
.LOC extended-pkt-address+?ELLN
```

Figure 2-116 shows the structure of the labeled tape extension.

If you create and open a labeled tape file with the ?OPEN creation option, the operating system treats the extension as follows:

- If you supplied values in the extension, the operating system uses those values.
- If you did not supply values in the extension, the operating system fills the packet extension with the default values listed in Table 2-94.

Mountstatus Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-124 must contain the byte address of the queue name buffer for the mountstatus function. To obtain all the mount queue names, use the ?XFNQN (get queue names) function of system call ?EXEC.

Figure 2-152 shows the structure of the mountstatus command subpacket, and Table 2-131 describes its contents.

	0	15 16	31	
?ZMSI	Subpacket identifier			
?ZMSF	Flags -- 1	Flags -- 2		?ZMSG
?ZMSP	Requestor PID	EXEC son PID		?ZMSE
?ZMSK	Get next key -- 1			
?ZMSX	Get next key -- 2			
?ZMSU	Byte pointer to user name buffer			
?ZMS3	Number of bytes in the username buffer	Number of bytes in the string returned in the username buffer		?ZMS4
?ZMST	Byte pointer to text name buffer			
?ZMS5	Number of bytes in the text name buffer	Number of bytes in the string returned in the text name buffer		?ZMS6
?ZMSV	Byte pointer to void name buffer			
?ZMS7	Number of bytes in the void name buffer	Number of bytes in the string returned in the void name buffer		?ZMS8
?ZMSD	Byte pointer to device name list buffer			
?ZMS9	Number of bytes in the device name list buffer	Number of bytes in the string returned in the device name list buffer		?ZMS0
?ZMSS	Byte pointer to void list name buffer			
?ZMSA	Number of bytes in the void list name buffer	Number of bytes in the string returned in the void list name buffer		?ZMSZ
?ZMSM	Mount ID	Mount error		?ZMSR
	?ZMSL = packet length			

Figure 2-152. Structure of Mountstatus Command Subpacket

?OPEX Continued

Table 2-131. Contents of Mountstatus Command Subpacket

Offset	Contents
?ZMSI (double-word)	Subpacket identifier. Place ?ZMSQ here.
?ZMSF	Flags word. It contains bit flags that indicate default choices to the operating system. You must set either bit ?ZY20 or bit ?ZY21. ?ZY20 = 1B(?ZZ20) -- You want the status of all requests, both inactive and active. ?ZY21 = 1B(?ZZ21) -- You want the status by mount ID. ?ZJ22 = 0B(?ZZ22) -- An output bit; explicit labeled mount request. ?ZY22 = 1B(?ZZ22) -- An output bit; implicit labeled mount request. ?ZY23 = 1B(?ZZ23) -- An output bit; unit mount request. ?ZY24 = 1B(?ZZ24) -- An output bit; waiting to be dismounted. ?ZY25 = 1B(?ZZ25) -- An output bit; next volume. ?ZY26 = 1B(?ZZ26) -- An output bit; mount error. ?ZY27 = 1B(?ZZ27) -- An output bit; specific volume. ?ZY28 = 1B(?ZZ28) -- An output bit; extend valid list. ?ZY29 = 1B(?ZZ29) -- An output bit; read only.
?ZMSG	Flags word. It contains bit flags that the operating system sets. ?ZY2G = 1B(?ZZ2G) -- An output bit; density is 800 bpi. ?ZY2H = 1B(?ZZ2H) -- An output bit; density is 1600 bpi. ?ZY2I = 1B(?ZZ2I) -- An output bit; density is 6250 bpi. ?ZY2J = 1B(?ZZ2J) -- An output bit; automatic density matching.
?ZMSP	The operating system returns the PID of the requestor.
?ZMSE	The operating system returns the PID of the immediate son of EXEC.
?ZMSK (double-word)	Get next entry key. The operating system uses this offset to hold the key indicator for EXEC.
?ZMSX (double-word)	Second get next entry key. The operating system uses this offset to hold the second key indicator for EXEC.

(continued)

?PRDB [*packet address*]

error return

normal return

?PWRB [*packet address*]

error return

normal return

Input

AC0	Reserved (Set to 0.)
AC1	Target file's channel number
AC2	Address of the packet, unless you specify the address as an argument to the system call

Output

AC0	Undefined
AC1	Byte count of the bytes read or written
AC2	Address of the packet

Error Codes in AC0

ERVWP Invalid word pointer passed as a system call argument

ER_FS_DIRECTORY_NOT_AVAILABLE

Directory not available because the LDU was force released (AOS/VS II only)

ER_FS_TLA_MODIFY_VIOLATION

Attempt to modify an AOS/VS II file with ?ODTL value supplied in ?GOPEN packet (?PWRB only)

Why Use It?

You can use ?PRDB or ?PWRB to check for bad blocks on a disk, or to check for problems with an I/O device.

When the operating system encounters a transfer error (bad block) while it is executing a ?PRDB or a ?PWRB, it takes the normal return and reports the reason for the error in the packet, but transfers all or part of that bad block. (See Table 2-159.) However, when a device error occurs during a ?PRDB or a ?PWRB, the operating system immediately aborts the transfer and returns the device error code to the packet.

Who Can Use It?

There are no special process privileges needed to issue this call. You must have obtained a channel number via ?GOPEN or ?OPEN before issuing this call. Also, you have Read access to the file before issuing ?PRDB or Write access to the file before issuing ?PWRB.

What It Does

?PRDB and ?PRWB read and write physical blocks on disk, respectively. Before you issue one of these system calls, load AC1 with the channel number assigned to the target file at ?GOPEN time.

?PRDB and ?PWRB each require an extended packet. You can specify the packet address as an argument to the system call or as input to AC2 before you issue the system call. The parametric value for the length of the packet, including the extension, is ?PPBLT. Figure 2-178 shows the structure and Table 2-157 describes the contents of the ?PRDB/?PRWB packet. Table 2-158 lists the disk and tape types applicable to the offsets in the packet extension.

?PRDB/?PWRB Continued

	0	15 16	31	
?PSTI	Status Word		Reserved (Set to 0.)	?PSTO
?PCAD	Word address of data buffer in your logical address space			
?PRNH	Block number of first block to be transferred			
?PRCL	Number of bytes in last disk or tape block transferred	Relative block number of last block transferred (disk) or running byte count (tape)		?PRBB
?PCS1	Controller status word 1	Controller status word 2		?PCS2
?PCS3	Controller status word 3	Controller status word 4		?PCS4
?PCS5	Controller status word 5	Controller status word 6		?PCS6
?PCS7	Controller status word 7	Controller status word 8		?PCS8
?PPBLT = packet length				

Figure 2-178. Structure of ?PRDB/?PWRB Packet

?PRDB/?PRWB Packet Offsets

Specify the tape or printing control options you want in the left byte of ?PSTI. Specify the number of blocks you want to read or write in the right byte of offset ?PSTI (block count). Use offset ?PCAD as a word pointer to the address of the data buffer you have reserved in your logical address space for the transfer. Offset ?PRNH must indicate the relative block number of the first block you want to transfer.

Use offset ?PRCL for ?PWRB (write block) only. Offset ?PRCL must contain the number of bytes in the last block you are writing. In effect, this value specifies the last valid byte in the block. If you are performing a write (?PWRB) to extend the file, the operating system places the end-of-file mark immediately after this byte. If you set offset ?PRCL to 0, the operating system sets the bytes in the last block to the default, which is 512 (a full block).

As Table 2-157 and Table 2-158 indicate, the operating system uses the ?PRDB/?PRWB packet extension to return status information about the block transfer. (You supply input values only for offsets ?PSTI through ?PRCL in the main packet.)

Table 2-157. Contents of ?PRDB/?PWRB Packet

Offset	Input Value	Output Value
?PSTI	Left Byte: Input options. ?IMIO = 1B(?IDIO) Use direct I/O with the MCA Protocol. The system ignores the block count, and transmits the number of bytes in ?PRCL. ?ENOV = 1B(?ENOR) For printers, enable a Vertical Form Unit load (part of Forms Control Utility). For tape, override the LEOT (logical end-of-tape) mark. ?SAFM = 1B(?SAFE) Enable a safe write to tape.	Unchanged.
	Right Byte: Block count.	Unchanged.
?PSTO	Reserved.	Unchanged.
?PCAD (doubleword)	Address of data buffer in your logical address space.	Unchanged.
?PRNH (doubleword)	Block number of first block to be transferred.	Unchanged.
?PRCL	Number of bytes in last disk or tape block transferred.	Unchanged.
?PRBB	Not applicable.	Number of blocks successfully transferred (disk) or running byte count (tape).
?PCS1 - ?PCS8	Not applicable.	Controller status words. (See Table 2-158.)

You can use offset ?PRBB to determine if an error was encountered, and if one was, the block number of the first bad block encountered during the transfer. Offset ?PRBB records the number of blocks that were successfully transferred. If ?PRBB does not contain the number you specified in offset ?PSTI, an error has occurred and offsets ?PCS1 through ?PCS8 in the packet extension record the reason for the transfer error.

Offsets ?PCS1 and ?PCS5 are always set on output, as appropriate for the device in use. Note that offsets ?PRBB through ?PCS5 are the only packet extension parameters currently in use.

?PRDB/?PWRB Continued

Table 2-158. ?PRDB/?PWRB Packet: Controller Status Words

DPD/DPG Disk	Status Word	Use
6030 (Floppy)	?PCS1	DIA (Transfer) Status
6045/6050/6051 (10 Mbytes)		
6070 (20 Mbytes)		
DPI Disk	Status Word	Use
6097 (Floppy)	?PCS1	DIA (Transfer) Status
6098/6099 (12.5 Mbytes)		
6100/6103 (25 Mbytes)		
6225 (5 Mbytes)		
6227 (15 Mbytes)		
6234 (50 Mbytes)		
DKB Disk	Status Word	Use
6063/6064/6065/6066 (Fixed Head)	?PCS1	DIA (Transfer) Status
	?PCS2	Unused
	?PCS3	ERCC Word One
	?PCS4	ERCC Word Two
DPF Disk	Status Word	Use
6060 (96 Mbytes)	?PCS1	DIA (Transfer) Status
6061 (147 Mbytes)	?PCS2	DIB (Drive) Status
6067 (50 Mbytes)	?PCS3	ERCC Word One
6122 (277 Mbytes)	?PCS4	ERCC Word Two
6160 (73 Mbytes)		
6161 (147 Mbytes)		
6214 (602 M bytes)		
DPM Disk	Status Word	Use
4514 (5 1/4 in. Floppy)	?PCS1	DIA (Transfer) Status
DPJ Disk	Status Word	Use
6236/6237 (354 Mbytes)	?PCS1	CB Status
6239/6240/ 6290/6350 (592 Mbytes)	?PCS2	Unit Status
6309 (5 1/4 in. Floppy)	?PCS3	CB Error Status
6310 (38 Mbytes)	?PCS4	Disk Error Code
6328 (70 Mbytes)	?PCS5	Controller Type
6329 (120 Mbytes)		

(continued)

Table 2-158. ?PRDB/?PWRB Packet: Controller Status Words

MTB/MTD Tape	Status Word	Use
6026 (800/1600 bpi)	?PCS1	DIA (Transfer) Status
4307 (1600/6250 bpi)	?PCS2	DIC (Drive) Status
6200 (1600/6250 bpi)		
6300 (1600/6250 bpi)		
MTC Tape	Status Word	Use
6125 (1600 bpi)	?PCS1	DIA (Transfer) Status
6231/6311 (Cartridge)		
MRC Disk	Status Word	Use
6236/6237 (354 Mbytes)	?PCS1	Controller Type
6239/6240/6290 (592 Mbytes)	?PCS2	Unit Type
6357/6398/	?PCS3	Controller Error Code
6399/6400 (862 Mbytes)	?PCS4	Error Status (ctrl/unit)
6581/6582/6584 (500 Mbytes)	?PCS5	Unit Status
	?PCS6	Drive Error Code
MRC Tape	Status Word	Use
6299/6300 (6250/1600 bpi)	?PCS1	Controller Type
4307-TL (6250/1600/800 bpi)	?PCS2	Unit Type
	?PCS3	Controller Error Code
	?PCS4	Error Status (ctrl/unit)
	?PCS5	Unit Status
	?PCS6	Drive Error Code

(concluded)

Table 2-159. Error Reports Returned in ?PRDB/?PWRB Offsets

Offsets	Contents	Meaning
?PSTI	20	
?PRBB	20	The operating system transferred all blocks without error.
?PSTI	20	
?PRBB	19	Last block was not transferred successfully. You might want to try again.

Notes

- Refer to the *ECLIPSE® MV/Family (32-Bit) Systems Principles of Operation* manual for details on the DIA, DIB, and DIC I/O instructions. Or, refer to the programmer's reference manual for your particular model.

?PRI

Changes the priority of the calling task.

?PRI

error return

normal return

Input

AC0 Priority number you wish to assign to the calling task

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Undefined

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?PRI is analogous to ?IDPRI, except that it changes the priority of the calling task, rather than another target task. Like ?IDPRI, ?PRI gives you some control over the operating system's task-scheduling activities. For example, you might use ?PRI to favor the calling task by assigning it a higher relative priority or, conversely, to favor other tasks by giving the calling task a lower relative priority.

Who Can Use It?

There are no special process privileges needed to issue this call, and there are no restrictions concerning file access.

What It Does

?PRI changes the priority number of the calling task to the priority that you specify in AC0. If there are already other tasks at the new priority level, the calling task ranks last in the priority queue and, therefore, gains CPU access after all others at that priority level. Like ?IDPRI, ?PRI can cause immediate task rescheduling.

Before you issue ?PRI, load AC0 with the priority number that you want to assign to the calling task. Priority numbers for tasks range from 0 (the highest priority level) through 255 (the lowest priority level). If you specify a priority number greater than 255, the operating system truncates the priority number to its least significant 8 bits.

Who Can Use It?

Although there are no special process privileges needed to issue this call, many options affect the privileges the newly created process has. For example, setting bit ?PFDA in offset ?PFLG prevents the operating system from passing the father's default ACL to the son. These options appear throughout the rest of this explanation of ?PROC.

You must have Execute access to the directory in which the program file resides along with both Execute and Read access to the program file itself.

What It Does

?PROC creates a process and assigns it whatever characteristics you specify in the ?PROC packet. Before you issue ?PROC, set up a packet in your logical address space that is ?PLTH words long. You can either cite the packet address as an argument to ?PROC, or you can load the packet address into AC2 before you issue ?PROC. Figure 2-179 shows the structure of the ?PROC packet and Table 2-160 defines each offset and mask.

Among other characteristics, ?PROC defines the process's type and priority, its maximum and minimum working set size, and its creation privileges (that is, the right to create sons with additional ?PROC system calls). Note that you cannot create a son process with privileges the calling process does not have.

Note also that when the operating system terminates a father process, it simultaneously terminates all his sons.

The ?PROC extension packets let you specify the following items when you issue ?PROC:

- CPU selection.
- User locality.
- ?PROC-related username.
- ?GROUP access control list (AOS/VS II only).

Figures 2-180 and 2-181 show the structure of the ?PROC extension packets, and Tables 2-162 and 2-163 define each offset.

Offset ?PFLG

Offset ?PFLG in the packet contains specification bits for the following characteristics and privileges:

- Process type.

Bit ?PFRS assigns the new process resident status; bit ?PFRP assigns it pre-emptible status. (Swappable is the default process type.) Bits ?PFRS and ?PFRP are mutually exclusive; if you set them both, the operating system takes the ?PROC error return and passes error code ERPTY to AC0.

?PROC Continued

- Default access control (AOS/VS only).

Bit ?PFDA, the access privileges bit, prevents the operating system from passing the father's file access privileges to the son. But, bit ?PFDA is only checked by the operating system if a username is specified, and if ?PUNM does *not* contain -1. If you set ?PFDA, the operating system assigns the son the default access control list USERNAME, OWARE (Owner, Write, Append, Read, Execute access). The son can also receive USERNAME OWARE access under the following conditions:

- When you assign the son a different username from its father.
- When the father process has no default access privileges.

For more information on default file access, see the description of ?DACL in this chapter.

- Control directive.

Bit ?PFDB directs the operating system to pass control to the Debugger utility immediately after it creates the new process. By default, control passes to the start of the program associated with the new process.

- Concurrency.

Bit ?PFEX blocks the calling process while the new process executes. If you don't set this bit, the son runs concurrently with its father. The calling process must have privilege ?PVEX to allow this. Under AOS/RT32, all processes have this privilege.

- Privileges mask.

Bit ?PFPM, the privileges mask, directs the operating system to give the son all the father's privileges except those specified in offset ?PPRV. Alternatively, if you set ?PFPM and set ?PPRV to -1, the son gets no privileges. If you just want the son to inherit the father's privileges, do NOT set ?PFPM, but set ?PPRV to -1.

- Break files and memory dumps.

On an AOS/VS system, setting ?PBRK and not ?PDMP creates a break file if the process traps. Not setting ?PBRK and setting ?PDMP creates a dump file of Ring 7 if the process traps. Setting both bits has exactly the same effect as not setting ?PBRK and setting ?PDMP.

On an AOS/RT32 system, all of the three combinations in the previous paragraph have the same effect. The effect is to create one break file with a memory dump of Ring 7. The filename has the format ?PID.TIME.7.BRK where PID and TIME are as described in Table 2-13 under offset ?ENFNP. The operating system creates another break file for each additional user ring (4, 5, 6) that is defined. A possible filename in this additional case is ?00035.11_41_23.6.BRK.

Table 2-161 describes each of the privilege bits in offset ?PPRV.

	0	15 16	31	
?PFLG	Process creation specs. (see Table 2-160)		Son process's priority	?PPRI
?PSNM	Byte pointer to program pathname			
?PIPC	Address of IPC message header			
?PNM	Byte pointer to son's simple process name			
?PMEM	Maximum number of logical pages			
?PDIR	Byte pointer to working directory name			
?PCON	Byte pointer to name of @CONSOLE device			
?PCAL*	Max. no. of system calls	Maximum working set size		?PWSS
?PUNM	Byte pointer to son's username			
?PPRV	Son's privileges	Maximum number of sons this son can create		?PPCR
?PWMI	Minimum working set size	Reserved (Set to 0.)		
?PIFP	Byte pointer to pathname of @INPUT file			
?POFP	Byte pointer to pathname of @OUTPUT file			
?PLFP	Byte pointer to pathname of @LIST file			
?PDFP	Byte pointer to pathname of @DATA file			
?SMCH	Maximum CPU time			
?PLTH = packet length				

* This offset differs between AOS/V5 and AOS/RT32. See Table 2-160.

Figure 2-179. Structure of ?PROC Packet

?PROC Continued

Table 2-160. Contents of ?PROC Packet*

Offset	Contents
?PFLG	<p>Process creation specifications.</p> <p style="padding-left: 40px;">?PFPP--Reserved - set to 0.</p> <p>Control directive.</p> <p style="padding-left: 40px;">?PFDB--Control goes to the Debugger.</p> <p style="padding-left: 40px;">DEFAULT = 0 (control goes to the starting address of the program file).</p> <p>Concurrency.</p> <p style="padding-left: 40px;">?PFEX--Caller is blocked while son executes.</p> <p style="padding-left: 40px;">DEFAULT = 0 (son runs concurrently with the ?PROC caller).</p> <p>Privileges mask.</p> <p style="padding-left: 40px;">?PFPM--Mask son's privileges.</p> <p style="padding-left: 40px;">DEFAULT = 0 (son process receives all privileges selected in ?PPRV).</p> <p>Access privileges.</p> <p style="padding-left: 40px;">?PFDA--Do not pass default ACL. ?PFDA is only checked if a username is specified, and ?PUNM does not contain -1.</p> <p style="padding-left: 40px;">DEFAULT = 0 (son has same default ACL as father (the caller)).</p> <p>Break file.</p> <p style="padding-left: 40px;">?PBRK--Create a break file if process traps or terminates fatally.</p> <p style="padding-left: 40px;">DEFAULT = 0 (do not create a break file if the process traps or terminates fatally).</p> <p>Block the son process.</p> <p style="padding-left: 40px;">?PFBS--Block son.</p> <p style="padding-left: 40px;">DEFAULT = 0 (do not block the son).</p> <p>Extension packet.</p> <p style="padding-left: 40px;">?PFXP--An extension packet exists.</p> <p style="padding-left: 40px;">DEFAULT = 0 (no extension packet exists).</p>

* There is no default unless otherwise specified. (continued)

Table 2-160. Contents of ?PROC Packet*

Offset	Contents
?PFLG (continued)	<p>Process type</p> <p>?PFRP--pre-emptible. ?PFRS--resident.</p> <p>DEFAULT = 0 (son is a swappable process).</p> <p>Process traps.</p> <p>?PDMP--Dumps Ring 7. (If ?PBRK is also set, overrides ?PBRK.)</p>
?PPRI	<p>Priority of the son process.</p> <p>DEFAULT = -1 (same as caller's priority).</p>
?PSNM (doubleword)	<p>Byte pointer to pathname of program file for the new process to execute.</p>
?PIPC (doubleword)	<p>Address of an IPC message header; this message is sent to the new process.</p> <p>DEFAULT = -1 (no message header).</p>
?PNM (doubleword)	<p>Byte pointer to new process's simple process name.</p> <p>DEFAULT = -1 (Son's simple process name is the ASCII representation of its PID (five digits, including leading zeros).)</p>
?PMEM (doubleword)	<p>Maximum number of logical pages in new process.</p> <p>DEFAULT = -1 (same logical page maximum as the caller's).</p>
?PDIR (doubleword)	<p>Byte pointer to name of son's working directory; if 0, then use the caller's current working directory.</p> <p>DEFAULT = -1 (same initial working directory as the caller's).</p>

* There is no default unless otherwise specified.

(continued)

?PROC Continued

Table 2-160. Contents of ?PROC Packet*

Offset	Contents
?PCON (doubleword)	Byte pointer to name of new process's @CONSOLE file; if 0, there is no @CONSOLE file. DEFAULT = -1 (same @CONSOLE device as caller).
?PCAL	Maximum number of system calls the son can issue concurrently. DEFAULT = -1. (Son is limited to two concurrent system calls.) AOS/RT32 ignores this offset; set to -1.
?PWSS	Maximum working set size for son. DEFAULT = -1 (no limit on working set size; otherwise, caller's limit). NOTE: The calling process must have the ?PVWS privilege to set this parameter.
?PUNM (doubleword)	Byte pointer to son's username. DEFAULT = -1 (same username as the caller's). NOTE: If ?PUNM contains -1 and a username is specified, ?PFDA is also checked.
?PPRV	Son's privileges; each bit in this offset (Table 2-161) assigns a privilege if ?PFPM is not set, or denies a privilege if ?PFPM is set. ?PFPM is a flag in offset ?PFLG (described earlier in this table). DEFAULT = -1. (If ?PFPM is also set, son has no privileges; if ?PFPM is not set, son has all the caller's privileges.)
?PPCR	Maximum number of sons this son can create; if 0, no sons. DEFAULT = -1 (same number of sons as the caller's (minus the number already created)).

* There is no default unless otherwise specified.

(continued)

Table 2-160. Contents of ?PROC Packet*

Offset	Contents
?PWMI	Minimum working set size. DEFAULT = -1 (no working set minimum; otherwise, caller's minimum). NOTE: The calling process must have ?PVWS privilege to set this parameter.
?PIFP (doubleword)	Byte pointer to pathname of son's @INPUT file; if 0, there is no @INPUT file. DEFAULT = -1 (same @INPUT file as the caller).
?POFP (doubleword)	Byte pointer to pathname of son's @OUTPUT file; if 0, there is no @OUTPUT file. DEFAULT = -1 (same @OUTPUT file as the caller).
?PLFP (doubleword)	Byte pointer to pathname of son's @LIST file; if 0, there is no @LIST file. DEFAULT = -1 (same @LIST file as the caller).
?PDFP (doubleword)	Byte pointer to pathname of son's @DATA file; if 0, there is no @DATA file. DEFAULT = -1 (same @DATA file as the caller).
?SMCH (doubleword)	Maximum CPU time allotted to the son. DEFAULT = -1 (Son receives remainder of father's time limit); if -1 and if father has no time limit, son has no time limit.

* There is no default unless otherwise specified. (concluded)

?PROC Continued

Table 2-161 describes each of the privilege bits in offset ?PPRV.

Table 2-161. Privilege Bits in Offset ?PPRV

Privilege	Meaning
?PVPC	The new process can create an unlimited number of sons.
?PVWS	The new process can ?PROC sons of a different program file type (that is, 16-bit or 32-bit program file).
?PVEX	The new process can remain unblocked while its sons execute.
?PVWM	The new process can define its sons' working set limit.
?PVPR	The new process can either assign itself a higher priority by issuing ?PRIPR or it can ?PROC a son with a higher priority than itself.
?PVTY	The new process can change its own process type (?CTYPE) or ?PROC a son with a different process type. (By default, sons have the same process types as their fathers.)
?PVUI	The new process can assign its sons different usernames.
?PVDV	The new process can define and access user devices.
?PVIP	The new process can issue the primitive IPC system calls ?ISEND and ?IS.R. (Note that connected processes, i.e., customer/servers, do not need this privilege to issue ?IS.R.)
?PVSM	The process can become System Manager.
?PVSU	The new process can issue ?SUSER to enter Superuser mode.
?PVSP	The new process can issue ?SUPROC to enter Superprocess mode.
?PVPP	The new process can be a peripheral process.

Offset ?PIPC

Offset ?PIPC points to an IPC message header. Frequently, you may want to send a CLI-format command line as the IPC message. This way, the son process can access arguments and switch values via a ?GTMES call. The explanation of system call ?GTMES explains a CLI-format line. In Appendix A sample programs BOOMER, DLIST, and TIMEOUT illustrate the ?GTMES call. Here's how to create such an IPC message:

- Set the system flag word (offset ?ISFL) to zero in the IPC message (as opposed to setting a bit if you were chaining to another process).
- Set bit ?RFCF in the user flag offset (?IUFL) to indicate CLI format.
- Set destination and origin port numbers to 0.
- Set offset ?ILTH to the word length of the message.
- Set offset ?IPTR to the word address of the message.

See Figure 2-83 for the structure of the ?ISEND header.

Suppose you issue ?PROC with offset ?PIPC containing the address of an IPC message header, and the target process will read the IPC message via ?GTMES. Then, offset ?IUFL of the IPC message header must contain ?RFCF to specify CLI format. Figure 2-83 and Table 2-60 describe the IPC message header; Figure 2-82 describes offset ?IUFL.

Other Offsets

Offset ?PPRI specifies a priority number for the new process, within the range 1 through 511. When you default this parameter (set offset ?PPRI to -1), the new process assumes its father's priority number. A father process must have privilege ?PVPR to assign to its son a higher priority than its own.

Offset ?PMEM specifies the maximum number of logical pages the new process can contain. By default, fathers and sons have the same logical page limit.

Offset ?PCAL specifies the maximum number of concurrent system calls the new process can issue. In general, critical processes should have higher ?PCAL parameters than less critical processes. Note that ?PCAL, and all specifications in ?PROC, should match the new process's needs as closely as possible.

Creating Offspring

The system determines the number of offspring that a process can create by checking its ?PROC packet for the following:

- The ?PVPC privilege in offset ?PPRV. This privilege allows the new process to create an unlimited number of sons. ?PVPC overrides the value in offset ?PPCR, described next.
- The ?PPCR offset which specifies the maximum number of offspring the new process can create. This offset is cumulative, which means that if a process with a ?PPCR value of 10 creates 2 sons, each with a ?PPCR value of 4, the original process cannot create any more sons, because 2 sons plus 8 (2*4) potential grandsons equals 10. This example is true as long as the 2 sons actually create 8 processes. If they actually create only 5 processes between them, the original process can create at most 3 processes (2 + 5 + 3 = 10).

?PROC Continued

- The ?PVEX bit in offset ?PPRV and the ?PFEX mask in offset ?PFLG. ?PVEX ensures that the new process remains unblocked while its sons execute (which allows it to create other sons). ?PFEX results in the ?PROC caller being blocked while the new process executes.

When a process which lacks the ?PVPC privilege tries to create a son, the operating system will create the son if

- The number of sons and their combined ?PPCR count (maximum number of sons which can be created) is less than or equal to the caller's ?PCPR value. (If the sum is more, the operating system signals an error.)
- Bit ?PVEX is set. (This lets the new process remain unblocked while its sons execute.)

Setting Maximum CPU Time

Offset ?SMCH specifies the maximum number of CPU milliseconds you want to assign the new process and all its subordinates. You must express this parameter as a double-precision unsigned integer. When a process exceeds its CPU time limit, the operating system terminates that process and returns error code ERTLM to its father.

When a process with a CPU time limit creates a son process, the operating system blocks the father and limits the son to the father's remaining CPU time. You can assign the son less than the father's remaining CPU time, but not more. If you assign more, the system takes the ?PROC error return and returns error code ERSMX (error in setting CPU time). As soon as the son terminates, the father receives whatever CPU time remains.

If you set offset ?SMCH to -1, the son acquires the rest of the father's CPU time. If you set offset ?SMCH to 0, the son has no CPU time limit, unless the father has a time limit. If the father has a time limit, the son gets the father's remaining time.

Setting the Working Set Size

Offsets ?PWMI and ?PWSS specify the new process's minimum and maximum working set size, respectively. Both values relate to the number of physical pages the process needs to execute efficiently. To set these values, the calling process must have privilege ?PVWS.

By default, the operating system adjusts the working set parameters dynamically, based on the process's type, page fault history, and general system overhead. When you set ?PWMI and ?PWSS, you direct the operating system to keep the working set within specific limits.

A process's working set size does not necessarily relate to the size of its logical address space. In fact, a larger process can require fewer physical pages than a smaller one if the larger process localizes its references and uses its databases efficiently. If you do set the working set size, take care not to set the maximum too low, because doing so increases paging I/O and processing time. Also, take care not to set the minimum working set size too high, because doing so can reduce efficient memory usage throughout your system.

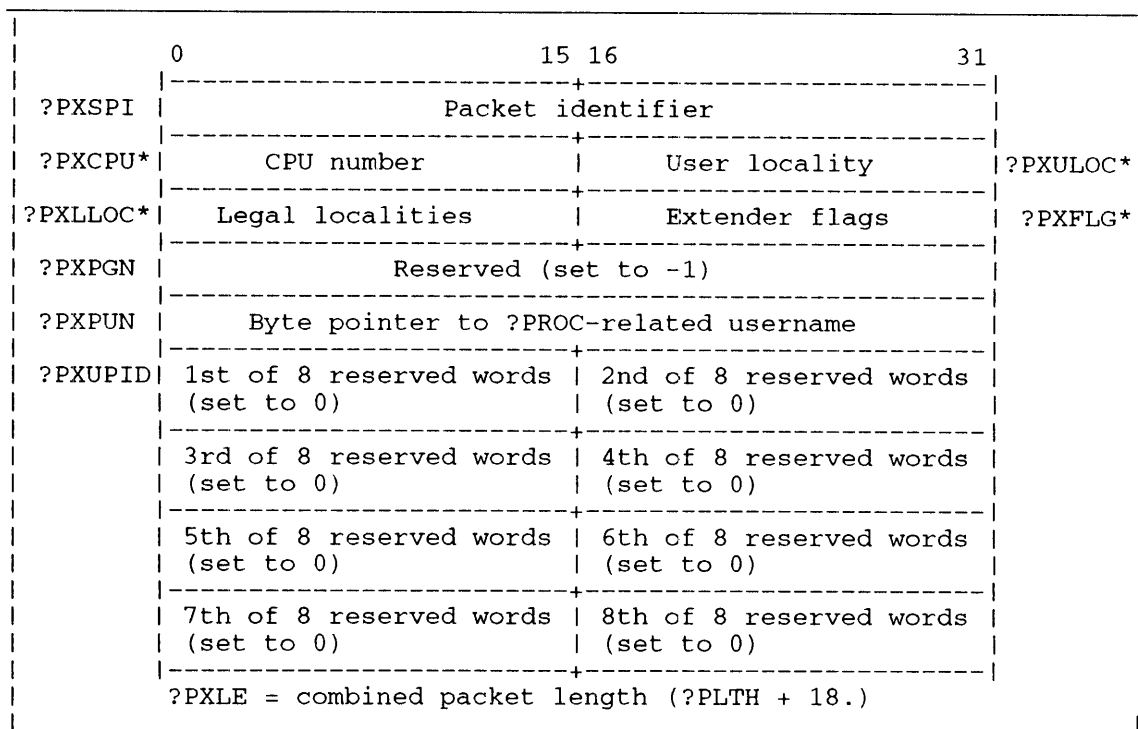
CAUTION: Generally, you will hurt the performance of the system if you define the size of a process's working set. Use this option with care; most of the time you'll be better off taking the default and allowing the operating system to make the necessary adjustments.

Extension Packet

You specify the existence of an optional extension packet by setting bit ?PFXP in offset ?PFLG of the main parameter packet. Then place the extension packet itself immediately after the end of the main parameter packet. The length of the combined packet is ?PXLE words.

Figure 2-180 contains the structure of the ?PROC extension packet for AOS/VS and AOS/RT32 and Table 2-162 describes its contents. Figure 2-181 contains the structure of the ?PROC extension packet for AOS/VS II and Table 2-163 describes its contents.

?PROC Continued



* This offset differs between AOS/VS and AOS/RT32. See Table 2-162.

Figure 2-180. Structure of ?PROC Extension Packet for AOS/VS and AOS/RT32

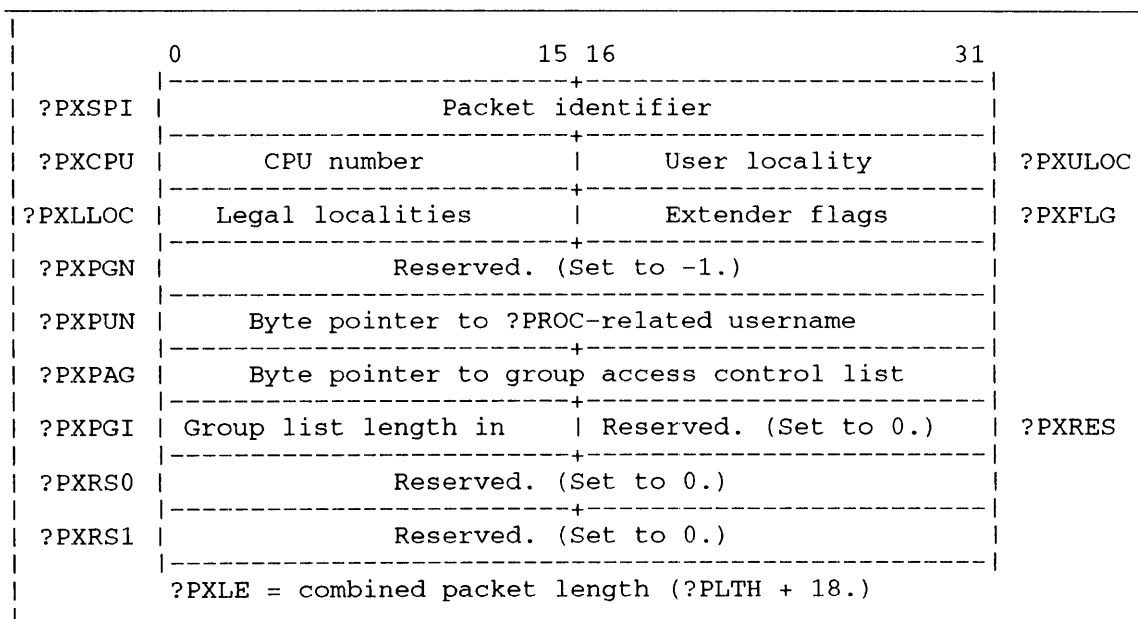


Figure 2-181. Structure of ?PROC Extension Packet for AOS/VS II

In Table 2-162, the column heads "VS" and "RT" represent AOS/VS and AOS/RT32, respectively. The entries in these columns have the following codes:

- The operating system does not define a value for the given offset.
- [no entry] The operating systems handle the offset in the sameway.

Table 2-162. Contents of ?PROC Parameter Packet Extension for AOS/VS and AOS/RT32

Offset	VS	RT	Description
?PXSPI (dbl wd)			Packet identifier. Place 0 here.
?PXCPU		---	CPU number. Place here the ID of the CPU on which this process executes. AOS/RT32 does not use this value; supply -1.
?PXULOC		---	Place here the user locality as you have defined it. AOS/RT32 does not use this value; supply -1.
?PXLLOC		---	Legal user localities. This offset indicates the legal user localities that the new process may change to. The format is a bit map. If a bit is set, the new process may change to the corresponding user locality. Note that this format is the same one that ?PROFILE returns, so you can take the legal user localities from the profile and insert them directly into this extension packet. AOS/RT32 ignores this offset; supply -1.
?PXFLG		---	Extender flags word. This contains bit ?PFL. If the bit is set, AOS/VS uses a default value for the son's legal locality. In other words, AOS/VS would NOT refer to offset ?PXLLOC to get the son's locality. If the bit is not set, AOS/VS obtains the son's locality from offset ?PXLLOC. AOS/RT32 does not use this information; supply -1.
?PXPGN (dbl wd)			Reserved. (Set to -1.)
?XPUN (dbl wd)			Byte pointer to the ?PROC-related username that the new process will be created with, but only if the process has Superprocess privilege and has turned it on. Otherwise, supply -1. Normally only Data General software such as EXEC supplies a byte pointer here.
?PXUPID (8 words)			Reserved. (Set these 8 words to 0.)

?PROC Continued

Table 2-163. Contents of ?PROC Parameter Packet Extension for AOS/VS II

Offset	Description
?PXSPI	Packet identifier; ?PXSID defines the packet for AOS/VS II use.
?PXCPU	CPU number. Place the ID of the CPU on which this process executes.
?PXULOC	Specify the user locality.
?PXLLOC	Legal user localities. This offset indicates the legal user localities to which the new process may change. The format is a bit map. If a bit is set, the new process may change to the corresponding user locality. Note that this format is the same one that ?PROFILE returns, so you can take the legal user localities from the profile and insert them directly into this extension packet.
?PXFLG	Extender flags word. This word contains bit ?PFDLL. If the bit is set, AOS/VS uses a default value for the son's legal locality. In other words, AOS/VS II would NOT refer to offset ?PXLLOC to get the son's locality. If the bit is not set, AOS/VS II obtains the son's locality from offset ?PXLLOC.
?PXPGN	Reserved. (Set to 0.)
?PXPUN	Byte pointer to the ?PROC-related username that the new process creates, but only if the process has Superprocess privilege turned on. Otherwise, supply -1. Normally only Data General software such as EXEC supplies a byte pointer here.
?PXPAG	Byte pointer to the group access control list. A group access control list is a double, null-terminated list of group names. A group name is a null-terminated, ASCII string with a maximum length of 16 bytes, including the null character. A group name corresponds to a filename in the :GROUPS directory. If you specify -1, use the parent's group access control list. If you specify 0, use a null group access control list. If you call ?PROC for AOS/VS II with no extension packet or with the AOS/VS and AOS/RT32 extension packet, the call behaves as if you specified a -1 in ?PXPAG (process inherits its parent's list of groups).
?PXPGI	Length of the group access control list.
?PXRES	Reserved. (Set to 0.)
?PXRES0	Reserved. (Set to 0.)
?PXRES1	Reserved. (Set to 0.)

?PFFD — (doubleword) field descriptor, continued.

- ?PBATCHP — batch priority, 1 word.
- ?PCNSPRV — terminal usage privilege, 1 bit.
- ?PBCHPRV — batch usage privilege, 1 bit.
- ?PMODPRV — modem usage privilege, 1 bit.
- ?PVCNPRV — virtual terminal usage privilege, 1 bit (negative logic).
- ?PRRAPRV — remote resource access privilege, 1 bit (negative logic).
- ?PPWDPRV — change password privilege, 1 bit (negative logic).
- ?PNCRYPT — system password encrypted indicator, 1 bit.
- ?PMGSYS — system manager privilege, 1 bit.
- ?PCOMMNT — user comment, 80 bytes including a null byte.
- ?PBWSS — batch working set size, 2 words; the internal format is
word 0 — maximum
word 1 — minimum
- ?PBLMEM — batch logical memory, 2 words; the internal format is
word 0 — low order
word 1 — high order
- ?PNBWSS — nonbatch working set size, 2 words; the internal format is
word 0 — maximum
word 1 — minimum
- ?PNBLMEM — nonbatch logical memory, 2 words; the internal format is
word 0 — low order
word 1 — high order
- ?PMYSONS — many sons privilege, 1 bit.
- ?PCHTYP — change type privilege, 1 bit.
- ?PCHPRI — change priority privilege, 1 bit.
- ?PPDPMGR — define PMGR privilege, 1 bit.
- ?PPRNBLK — PROC no-block privilege, 1 bit.
- ?PCHUSER — change username privilege, 1 bit.
- ?PACDEV — access devices privilege, 1 bit.
- ?PUIPCS — use IPC privilege, 1 bit.
- ?PSUSER — Superuser privilege, 1 bit.
- ?PPSUPP — Superprocess privilege, 1 bit.

?PROFILE Continued

?PFFD — (doubleword) field descriptor, continued.

- ?PWSON — wide son privilege, 1 bit.
- ?PMCTS — (read-only field) memory constraints, 10 words; the internal format is

words 0–1	?PBWSS
words 2–3	?PBLMEM
words 4–5	?PNBWSS
words 6–7	?PNBLMEM
word 8	?PSOPIO
word 9	?PBATCHP

- ?PHRDPRV — (read-only field) hard privileges, 1 word; the internal format is

bits 0–2	0
bit 3	?PCHWSSL
bit 4	?PWSON
bit 5	?PPSUPP
bit 6	?PMGSYS
bit 7	?PUIPCS
bit 8	?PACDEV
bit 9	?PCHUSER
bit 10	?PPRNBLK
bit 11	?PPDPMGR
bit 12	?PCHPRI
bit 13	?PSUSER
bit 14	?PCHTYP
bit 15	?PMYSONS

- ?PSFTPRV — (read-only field) soft privileges, 1 word; the internal format is

bits 0–9	0
bit 10	?PPWDPRV (negative logic)
bit 11	?PRRAPRV (negative logic)
bit 12	?PVCNPRV (negative logic)
bit 13	?PMODPRV
bit 14	?PBCHPRV
bit 15	?PCNSPRV

- ?PPRCINF — (read-only field) PROC information, 13 words; the internal format is

word 0	?PMXSONS
word 1	?PSFTPRV
word 2	?PHRDPRV
words 3–12	?PMCTS

- ?PINTDIR — initial directory, less than or equal to 64 bytes including a null byte.
- ?PCHWSSL — change WSS limit privilege, 1 bit.

Index

Within the index, a bold page number indicates a primary reference. A range of page numbers indicates the reference spans those pages. A reference such as ?RTODC_PKT... indicates that all references beginning with those characters are found on the referenced pages.

Symbols

! operator, 1–4
? CLI macro, 2–217
?CID_PKT.PKT_ID offset, 2–58.7–2–58.25
?ID7 value, 2–410
* and ** symbols, v, xi
< and > symbols, v, xi

Numbers

16-bit process system calls (names of), 2–14

A

AC0–AC3, 1–2–1–4
access
 control list, 2–77, 2–174, 2–240, 2–245, 2–650, 2–882
 shared, 2–699
 to a protected file, permitting, 2–519
 to all devices
 disabling, 2–81
 enabling, 2–83
 to memory, read/write, 2–782
accumulators, 1–2, 1–5
ACK0 and ACK1 characters, 2–674
ACL
 changing a file's group, 2–240
 getting a file's group, 2–882
 getting a file's, 2–77, 2–174, 2–245, 2–650
acquiring
 a new resource, 2–328
 resource, 2–594
active
 group of windows, 2–574
 PIDs, returning, 2–217
Ada language, 1–10
address
 logical, 2–781

request
 for consoles, 2–58.6
 for terminals, 2–58.6
ring base, 2–565
space
 logical, mapping a device into, 2–373
 remapping a process's, 2–353
?AEPR value, 2–618
Agent, changing the wiring characteristics, 2–18
?ALAU value, 2–618
?ALLOCATE system call, 2–15–2–20, 2–213
allocated blocks, reading, 2–23
allocating, disk blocks, 2–15
ANSI-standard terminal, 2–182
?AOPR value, 2–618
AOS/RT32
 and old file system, 2–205
 system calls names of, –3
AOS/VS
 and old file system, 2–205
 system resources system calls (names of), 2–12
AOS/VS II, and new file system, 2–205
?APND value, 2–408, 2–412
?ASEB value, 2–618
assembly language, iv, 1–1, x
 programming, 1–5–1–9
 sample program sets, iv, x
?ASSIGN system call, 2–17
attribute, permanent file, 2–653
?AUAL value, 2–618
autobaud matching, 2–198
auxiliary clock, 2–401
?AWENT value, 2–18
?AWIRE system call, 2–18
?AWUDS value, 2–18

B

- B operator, 1–4
- ?B32N offset, 2–20
- ?BADL and ?BADR offsets, 2–20
- base
 - address, ring, 2–565
 - current resource, 2–186
- BASIC language, 1–10
- baud, autobaud matching, 2–198
- baud rate, split baud, 2–197
- ?BBAC offset, 2–20
- ?BBAL offset, 2–20
- ?BBAN offset, 2–20
- ?BBLC offset, 2–20
- ?BBL offset, 2–20
- ?BBLN offset, 2–20
- BCC character, 2–674
- ?BCHN offset, 2–20
- ?BERR offset, 2–20
- bias factor values
 - getting, 2–176
 - setting, 2–655
- binary
 - mode, 2–40
 - synchronous communications line, opening a,
2–405
- bisecond, 2–74
- bit
 - setting a, 1–4
 - significant, 1–5
- ?BITM value, 2–376
- ?BLBB offset, 2–20
- ?BLKIO system call, **2–19–2–20**, 2–213, 2–216,
2–596
- ?BLKPR system call, 2–26
- bloc, disk identification, 2–291
- block
 - output to a data channel line printer,
2–600–2–601
 - reading allocated, 2–23
- block I/O, 2–19, 2–183, 2–210
 - disk, 2–598
 - MCA, 2–599
 - opening a file for, 2–210, 2–405
 - performing, 2–19, 2–596
 - physical, 2–24–2–25, 2–210, 2–525, 2–592
 - reading, 2–596
 - tape, 2–599
 - writing, 2–596, 2–844
- blocking a process, 2–26
- ?BLTH value, 2–20
- ?BM32R value, 2–21
- ?BMAFE value, 2–21
- ?BMBI value, 2–115–2–116, 2–117
- BMC device, 2–151, 2–271
- ?BMDBA value, 2–116
- ?BMDEV offset, 2–198
- ?BMDEV value, 2–191
- ?BMDIO value, 2–21
- ?BMEOR value, 2–21
- ?BMEP value, 2–115
- ?BMFO value, 2–117
- ?BMIO value, 2–21
- ?BMNAB value, 2–21
- ?BMNH and ?BM8B values, 2–115
- ?BMNMB value, 2–21
- ?BMNO value, 2–117
- ?BMNR value, 2–116
- ?BMOP value, 2–117
- ?BMPE value, 2–116
- ?BMPIO value, 2–21
- ?BMRA value, 2–117–2–118
- ?BMSH value, 2–116
- ?BMTB value, 2–115
- ?BMTI value, 2–117–2–118
- ?BMUC value, 2–117
- ?BNAME system call, 2–28
- BOOMER.SR sample program, A–2,
A–32–A–36
- boot clock, SCP, 2–401
- ?BPEL offset, 2–20
- ?BPER offset, 2–20
- ?BPVB offset, 2–20
- ?BR0BIT–?BR4BIT, baud rate offsets, **2–196**
- BRAC0–BRAC3 status words, 2–30
- BRAN utility program, 2–382

break

- connection, 2-516, 2-523
- customer/server relationship, 2-80, 2-92
- file
 - creating a, 2-29
 - disabling a, 2-95
 - enabling a, 2-94
- line, 2-40
- sequences, **2-198**

?BRFCT offset, 2-197

BRFP status word, 2-30

?BRKFL system call, 1-2, 2-29-2-30

BRPC status word, 2-30

BRSB status word, 2-30

BRSL status word, 2-30

BRSP status word, 2-30

BRTID status word, 2-30

BSC

- error statistics, 2-686
- line
 - device name for, 2-669
 - enabling a, 2-669
 - receiving information over a, 2-709
 - sending information over a, 2-720
- protocol data-link control characters, 2-674

?BSTS offset, 2-20

?BTBC offset, 2-20

?BTBL offset, 2-20

buffer

- mode of tape I/O, 2-214, 2-412
- moving bytes from a customer, 2-377
- moving bytes to a customer, 2-379

byte pointer, 1-5

C

C language, 1-10

?C16B offset, 2-198

?C8BT value, 2-178, 2-181

?CABD offset, 2-198

?CACC offset, 2-198

?CACP offset, 2-62-2-68
example of, 1-7

calendar, system, 2-660

calling resource, 2-328

?CALLOUT offset, 2-198

calls, system, 1-1

canceling, character device, 2-82

carriage return character, 2-663

cascaded virtual timer, 2-794

?CBK0 offset, 2-198

?CBK1 offset, 2-198

?CBK2 offset, 2-198

?CCPS offset, 2-66-2-68, 2-168

?CCTD offset, 2-198

?CCTYPE offset, 2-199

?CDAY system call, 2-31

?CDEH offset, 2-66-2-68

?CDEL offset, 2-66-2-68

?CDRXON value, 2-40-2-41

?CDSBRK value, 2-40, 2-41

?CDT0-?CDT3 values, 2-179, **2-181**, 2-195

?CEB0 and ?CEB1 values, 2-179

?CEOC value, 2-179, 2-195

?CEOL value, 2-178

?CEOS value, 2-179, 2-194

?CEPI value, 2-178

?CESC value, 2-179

?CFF value, 2-178

?CFKT value, 2-180, 2-614

?CFTYP offset, 2-61-2-68
example of, 1-6

?CGNAM system call, 2-32

?CH4 offset, 2-180

CHAIN command, 2-255

?CHAIN system call, 2-33-2-34

chaining

- new procedure, 2-595
- programs, 2-33

changing

- agent wiring characteristics, 2-18
- priority of a process, 2-531
- priority of a task, 2-280, 2-530
- process type, 2-75
- unshared memory pages, 2-384
- user locality, 2-354
- working directory, 2-89

channel

- closing a, 2-38
- graphics output, 2-225
- map, data, 2-729
- number, 2-226, 2-405, 2-569
 - getting a file's complete pathname from, 2-32

character device, 2-177, 2-190
 assigning to a process, 2-17
 canceling or deassigning a, 2-82
 deassigning a, 2-82

characteristics
 another's, 2-191
 changing the Agent's wiring, 2-18
 current, 2-191
 current and default, 2-190, 2-679
 default, 2-191
 device, 2-177
 extended, 2-190
 extended device, 2-679
 owned, 2-191
 packet parameters, 2-193
 shared console, ownership, 2-198
 special keys and device types, 2-200

CHECK.F77 sample program, A-2, A-42

?CHFS offset, 2-64-2-65
 example of, 1-6

?CID_PKT.CHAN_NUM offset, 2-58.7-2-58.25

?CID_PKT.CON_LEN offset, 2-58.7

?CID_PKT.CON_PTR offset, 2-58.7

?CID_PKT.RBUF_PTR offset, 2-58.7

?CID_PKT.RBUF_LEN offset, 2-58.7

?CID_PKT.RDATA_LEN offset, 2-58.7

?CID_PKT.USER_FLGS offset, 2-58.7

?CID_PKT_LEN offset, 2-58.7

?CID_RET_TYPES offset, session connection types, 2-58.9

?CINT value, 2-711, 2-723

?CKVOL system call, 2-35

?CL_... offsets and values, 2-37

class
 assignments logical processor, 2-362
 IDs, 2-36
 matrix, 2-51
 process, 2-514
 scheduling
 statistics
 accumulating, 2-42
 returning, 2-45
 system calls (names of), 2-12

?CLASS system call, 2-36-2-56

clear to send modem option, 2-180

Clearing, LBUS interrupts, 2-272

clearing
 default access control list, 2-77
 device, 2-40
 execute-protection status, 2-141-2-142

?CLFP offset, 2-199

?CLI, macro, 2-217

CLI
 message, getting a, 2-250
 message format, 2-255
 program, 1-1

CLI-format
 command line, sending via ?PROC, 2-543
 IPC message, 2-307

CLI.PR, summary of, A-39

clip rectangle, 2-223

?CLMAX value, 2-180, 2-190

?CLMIN value, 2-190

?CLMSK data field length mask, 2-197

?CLN5 offset, 2-197

?CLN6 offset, 2-197

?CLN7 offset, 2-197

?CLN8 offset, 2-197

clock
 auxiliary, 2-401
 real-time, 2-313
 SCP boot, 2-401
 system, 2-187, 2-201, 2-258, 2-645, 2-732

?CLOSE system call, 2-38-2-56, 2-183
 example of, A-23

closing
 channel, 2-38
 file, 2-38, 2-183
 shared-access file, 2-658

?CLR DV system call, 2-40-2-41

?CLS_ACC value, 2-44

?CLS_GET value, 2-44

?CLS_PKT... offsets and values, 2-43-2-44

?CLS_SCHED value, 2-44

?CLS_SET value, 2-44

?CLSCHEM system call, 2-42-2-43

?CLST_... offsets and values, 2-47-2-50

?CLSTAT system call, 2-45-2-50

?CLT0 offset, 2-197

?CLT1 offset, 2-197

?CLTH value, 2-62, 2-64, 2-66
 example of, 1-7

- ?CMAT_... offsets and values, 2-53-2-55
- ?CMATRIX system call, 2-51
- ?CMD0P offset, 2-198
- ?CMIL offset, 2-64-2-68
 - example of, 1-7
- ?CMOD value, 2-179, 2-181, 2-657, 2-680
- ?CMPLT value, 2-59
- ?CMRI value, 2-178, 2-181, 2-657, 2-680
- ?CMRS offset, 2-64-2-66
 - example of, 1-7
- ?CMSH offset, 2-64-2-65
 - example of, 1-7
- ?CNAS value, 2-178, 2-181-2-182
- ?CNLX function, 2-192
- ?CNNL value, 2-180, 2-181, 2-680
- ?CNRM value, **2-179**, 2-682
- COBOL language, 1-10
- CODE, MASM macro, 2-101
- codes
 - and text error, returning, 2-683
 - error, 1-9
 - exception, 1-2
 - parametric, 1-3
- command line (CLI-format), sending via
 - ?PROC, 2-543
- Commands, format conventions, xi
- communication, operator/current process, 2-437
- complete pathname
 - getting a, 2-32, 2-205
 - returning generic file, 2-239
- COMSWITCH device, 2-40
- ?CON system call, **2-56**, 2-160
- ?CON_PKT.USER_FLGS offset, input values, 2-58.8
- conditional I/O, 2-220, 2-707
- ?CONFIG system call, **2-58-2-58.5**
- ?CONFIG_... function codes, 2-58.1-2-58.5
- ?CONFIG_... offsets and values, for current device route, 2-58.3
- ?CONFIG_RESET_... contents and values, channel rerouting, 2-58.4
- ?CONINFO
 - and connection errors, 2-58.9
 - for TCONS, 2-58.9
 - and Permanent Virtual Circuits (PVCs), 2-58.9
 - and teletype connections, 2-58.9
 - and Telnet connections, 2-58.9
 - and Xerox Network Services (XNS), connections, 2-58.9
 - ?CON_CON_... offsets, 2-58.12-2-58.13
 - ?CON_ITC_... offsets, 2-58.14-2-58.15
 - ?CON_PVC_... offsets, 2-58.16-2-58.17
 - subsockets, 2-58.18-2-58.25
 - ?CON_TCP_... offsets, 2-58.10
 - ?CON_TNET_... offsets, 2-58.13-2-58.14
 - ?CON_TSC_... offsets, 2-58.15-2-58.16
 - ?CON_XNS_... offsets, 2-58.11-2-58.12
 - console line numbers, 2-58.8
 - console types, 2-58.9
 - return packet types, 2-58.8
 - TermServer console, 2-58.9
- ?CONFINFO packet contents, 2-58.7
- ?CONINFO system call, **2-58.6-2-58.25**
- connect time-out, 2-677
- connection
 - breaking, passing, and re-establishing a, 2-516, 2-523
 - management, 2-57
 - message, 2-295
 - system calls (names of), 2-11, 2-13
 - session types, 2-58.9
- connections, Xerox Network Services (XNS), 2-58.9
- console
 - address request, 2-58.6
 - line numbers, 2-58.8
 - types, 2-199
- construction, program, 1-8
- ?CONT value, 2-711, 2-716, 2-723
- Contacting Data General, xii
- contacting Data General, vi
- continue/receive call, 2-714
- control
 - characters (data-link), BSC protocol, 2-674
 - passing from one program to another, 2-33
 - point directory
 - ?CREATE packet for, 2-64
 - maximum size of, 2-58.26
 - station, multipoint, 2-715
- control characters, effect
 - erase line, 2-200
 - move left, 2-200
 - move right, 2-200
 - rubout echo, 2-200
- CONTROL @EXEC family of commands, 2-438
- control keys, characteristics, 2-200
- control-character terminal interrupt
 - disabling, 2-333

- re-enabling, 2-334
- controller
 - intelligent, 2-190
 - status word, 2-528-2-529
- converting
 - date to a scalar value, 2-143
 - scalar date value, 2-31
 - scalar time value, 2-74
 - time of day to a scalar value, 2-171
- ?COTT value, 2-178
- ?CPBN value, 2-179, 2-181
- ?CPEN offset, 2-197
- CPI/24 device, 2-41
- ?CPM value, 2-179, 2-181
- ?CPMAX system call, **2-58.26**, 2-213
- ?CPMCN offset, 2-59
- ?CPMFW offset, 2-59
- ?CPMHS offset, 2-59
- ?CPMLS offset, 2-59
- ?CPMSK parity mask, 2-197
- ?CPOR offset, 2-62-2-63
- ?CPR1 offset, 2-197
- ?CPR2 offset, 2-197
- ?CPR0 offset, 2-197
- ?CPTY offset, 2-197
- CPU device, 2-151, 2-271
- CR character, 2-663
- ?CR110 offset, 2-196
- ?CR12H offset, 2-196
- ?CR134 offset, 2-196
- ?CR150 offset, 2-196
- ?CR18H offset, 2-196
- ?CR19K offset, 2-196
- ?CR20H offset, 2-196
- ?CR24H offset, 2-196
- ?CR300 offset, 2-196
- ?CR36H offset, 2-196
- ?CR38K offset, 2-196
- ?CR45 offset, 2-196
- ?CR48H offset, 2-196
- ?CR50 offset, 2-196
- ?CR600 offset, 2-196
- ?CR72H offset, 2-196
- ?CR75 offset, 2-196
- ?CR96H offset, 2-196
- ?CRAC value, 2-178
- ?CRAF value, 2-178
- ?CRAT value, 2-178
- ?CREATE system call, 1-1, 1-5, **2-60-2-68**, 2-168
 - example of, 1-6
- CREATE_WINDOW.SR sample program, A-2, **A-46-A-55**
- creating
 - break file, 2-29
 - directory or file, 2-60, 2-405, 2-849
 - dump file, 2-381
 - label for diskette or magnetic tape, 2-336
 - logical processor, 2-365
 - operator interface, 2-424
 - pipe file, 2-68
 - process, 2-534
 - queued task manager, 2-294
 - user data area, 2-70
- ?CRT1-?CRT15 values, 2-181
- ?CRUDA system call, **2-70**, 2-213
- ?CS10 offset, 2-197
- ?CS15 offset, 2-197
- ?CS20 offset, 2-197
- ?CSBDS value, 2-680
- ?CSBEN value, 2-680
- ?CSFF value, 2-178
- ?CSMK stop bit mask, 2-197
- ?CSPO value, 2-178
- ?CSRDS offset, 2-198
- ?CST value, 2-178
- ?CTCC offset, 2-199
- ?CTCD offset, 2-199
- ?CTDW offset, 2-199
- ?CTERM system call, 2-72-2-73
- ?CTHC offset, 2-199
- ?CTIM offset, 2-62-2-68
 - example of, 1-7
- ?CTLT offset, 2-199
- ?CTO value, 2-179, 2-734
- ?CTOD system call, 2-74
- ?CTSP value, 2-179, 2-181
- ?CTYPE system call, 2-75-2-76

- ?CUCO value, 2-178
- ?CULC value, 2-179, 2-181
- current
 - characteristics, 2-190, 2-679
 - date, 2-187, 2-313
 - process/operator process communication, 2-437
 - resource, base of the, 2-186
 - time, 2-313
- cursor hotspot, 2-573
- customer, 2-56
 - buffer
 - moving bytes from a, 2-377
 - moving bytes to a, 2-379
 - process, terminating a, 2-72
 - verifying a, 2-786, 2-789
- customer/server relationship, 2-34, 2-57
 - breaking a, 2-80, 2-92
 - terminating a, 2-72
- ?CWIN offset, 2-198
- ?CWIN value, 2-191
- ?CWRP value, 2-179
- ?CXLT offset, 2-198
- ?CXLT sets DG ANSI mode, 2-191

D

- ?DAC2 offset, 2-91, 2-748-2-749
- ?DACL system call, 1-2, 2-77-2-78
- ?DADID system call, 2-79
- data, password encryption, 2-590
- data channel
 - line printer, 2-70, 2-600-2-602
 - map, 2-729
- data compression
 - and native mode, 2-215
 - magnetic tape, 2-215
- Data Encryption Standard, 2-590
- data field length, mask, 2-197
- Data General, contacting, vi, xii
- data-link control characters, BSC protocol, 2-674
- data-sensitive files, default delimiters for, 2-663
- date
 - converting to a scalar value, 2-143
 - current, 2-187, 2-313

- last accessed value, 2-214
 - setting the, 2-400
 - value, converting a scalar, 2-31
- day
 - current, 2-187
 - time of, 2-258
- ?DCC offset, 2-91, 2-751-2-752
- DCH device, 2-151, 2-271
- ?DCI offset, 2-91, 2-751-2-752
- ?DCL2 offset, 2-748
- ?DCON system call, 2-80
- ?DDIS system call, 2-81
- dead space on a tablet, 2-572
- ?DEASSIGN system call, 2-82
- deassigning, character device, 2-82
- ?DEBL system call, 2-83
- ?DEBUG system call, 2-84
 - example of, A-16
- debugger, symbolic utility program, 1-3, 2-84
- debugging
 - program, 2-29-2-30
 - system calls (names of), 2-7
- decimal number specification, v, xi
- default
 - access control list, 2-77
 - characteristics, 2-190, 2-679
- defining
 - kill-processing routine, 2-330
 - poll-address pair, 2-664
 - polling list, 2-664
 - select-address pair, 2-664
 - terminal interrupt task, 2-293
 - user device, 2-269
 - fast, 2-149
- definition table, map, 2-153, 2-154, 2-272, 2-731
- ?DELAY system call, 2-85
- delaying, task, 2-85, 2-809
- ?DELETE system call, 2-86, 2-213
 - example of, A-19
- deleting
 - directory or file, 2-86
 - logical processor, 2-367
- delimiter table, 2-188, 2-416, 2-662
 - getting a, 2-188
 - setting a, 2-662
- delimiters for data-sensitive files, default, 2-663

density values, tape, 2-337

DES, 2-590

descriptor information, file, 2-780

device

- assigning to a process, 2-17
- character, 2-177, 2-190
- characteristics
 - commonly used, 2-180-2-202
 - complete, **2-194-2-199**
 - extended, 2-679
 - reading, 2-177
- clearing a, 2-40
- control table, 2-151, 2-271
- driver routine, 2-150, 2-270
- fast user, 2-149
- interrupt handler service routine, 2-156, 2-275
- mapping into logical address space, 2-373
- powerfail/restart routine, user, 2-277
- termination routine, 2-272
 - user, 2-276
- time-out value, 2-733-2-735
- user, 2-18, 2-269, 2-304.9

device reset, pending status, 2-58

devices

- disabling access to all, 2-81
- enabling access to all, 2-83

?DFLGS offset, **2-91**, 2-748-2-749

?DFLRC value, 2-749

?DFRSCH system call, 2-88

DG/VIEW windowing user interface, 2-828

DIB, 2-291

?DID offset, **2-91**, 2-748, 2-749

digitize option for a tablet, 2-572, 2-574, 2-579

?DIR system call, 1-5, 2-89

- examples of, 1-6, A-42

DIRCREATE.F77 sample program, A-2, **A-41-A-45**

DIRCREATE.SR sample program, 1-5-1-9

direct-access vertical forms control unit, 2-600-2-601

directory

- changing the working, 2-89
- creating a, 2-60
- deleting a, 2-86
- entries, 2-207
- file
 - ?CREATE packet for, 2-64
 - getting status of, 2-168-2-169
- filenames, 2-207

disabling

- access to all devices, 2-81
- break file, 2-95
- BSC line, 2-661
- class scheduling, 2-42
- control-character terminal interrupt, 2-333
- LEF mode, 2-350
- relative terminal, 2-667
- task rescheduling, 2-88
- task scheduling, 2-93
- terminal interrupt, 2-403

disconnecting, customer/server relationship, 2-80, 2-92

disk

- block I/O, 2-598
- blocks, allocating, 2-15
- identification bloc, 2-291
- initialized logical, 2-630
- logical, 2-630
- logical, initializing an extended, 2-886
- unit image, synchronized logical, 2-887

Disk Jockey utility, 2-213

diskette label, 2-336

display MRC routes, current, 2-58

DLCC, 2-674

DLE character, 2-674

DLE EOT characters, 2-674

DLIST.SR sample program, A-2, **A-26-A-27**

?DLNK offset, **2-91**, 2-748, 2-749

?DLNKB offset, **2-91**, 2-748, 2-749

?DLNKBL offset, 2-748

?DLNKL offset, 2-91

?DLNL offset, 2-748

?DNUM offset, **2-91**, 2-748, 2-749

Document sets, ix

documentation, related, iv, x

?DPC offset, **2-91**, 2-748, 2-749

?DPCL offset, 2-748

?DPRI offset, **2-91**, 2-748, 2-749

?DQTSK system call, 2-90

?DRCON system call, 2-92

?DRES offset, **2-91**, 2-748-2-749

?DRSCH system call, 2-93

DRT device, 2-41, 2-151, 2-271

?DSCH value, 2-88

?DSFLT offset, **2-91**, 2-748-2-749

?DSH offset, **2-91**, 2-751-2-752

- ?DSLTH value, 2-748
- ?DSMS offset, 2-91, 2-751-2-752
- DSR value, 2-677
- ?DSSL offset, 2-748-2-749
- ?DSSZ offset, 2-91, 2-748-2-749
- ?DSTB offset, 2-91, 2-748-2-749
- ?DSTL offset, 2-748
- DUART device, 2-151, 2-271
- DUMP CLI command, 2-110
- dump file, creating a, 2-381
- Dump Tool utility program, 2-382
- DUMP_II CLI command, 2-110
- dumping, a memory image, 2-381
- duplex, half, 2-198
- DVFU, 2-600
- .DWORD assembly language statement, 1-4
- ?DXLTH value, 2-91, 2-751

E

- ?EBAS value, 2-618
- ?EFP offset, 2-606, 2-617
- ?EFLN offset, 2-606, 2-617
- ?EFMAX value, 2-606, 2-617
- ?EFNF offset, 2-606, 2-617
- ?EFTL offset, 2-606, 2-617
- ?EFTY offset, 2-606, 2-617
- ?ELAC offset, 2-419, 2-421
- ?ELCR offset, 2-419-2-420
- ?ELCT offset, 2-419-2-420
- ?ELFS offset, 2-419, 2-421
- ?ELGN offset, 2-419-2-420
- ?ELL1-?ELL3 values, 2-421
- ?ELLN value, 2-419
- ?ELRE offset, 2-419-2-420
- ?ELUH offset, 2-419, 2-421
- ?ELUT offset, 2-419, 2-421
- ?ELVL offset, 2-419-2-420
- ?ELVR offset, 2-419-2-420
- .ENABLE assembly language statement, 1-4

- enabling
 - access to all devices, 2-83
 - break file, 2-94
 - BSC line, 2-669
 - class scheduling, 2-42
 - LEF mode, 2-351
 - multitask scheduling, 2-102
 - terminal interrupt, 2-404
- ?ENBFL offset, 2-95-2-96
- ?ENBLN value, 2-96
- ?ENBRK system call, 2-94-2-96
- encryption, password, 2-554, 2-590
- ?ENCST value, 2-96
- end-of-volume, forcing on labeled tape, 2-148
- ?ENDIR value, 2-96
- ?ENESH offset, 2-96
- ?ENET offset, 2-406-2-407, 2-411, 2-606-2-607
- ?ENEUS offset, 2-96
- ?ENFNP offset, 2-96
- ?ENID value, 2-672
- ?ENOV offset, 2-527
- ?ENOV value, 2-598
- ?ENPRE value, 2-96
- ENQ character, 2-675
- ?ENQUE system call, 2-98
- ?ENR4-?ENR7 values, 2-95-2-96
- ?ENSH value, 2-96
- ?ENSSH offset, 2-96
- ?ENSUS offset, 2-96
- entering
 - event in the system log file, 2-360
 - privilege state, 2-744
 - Superprocess mode, 2-735
 - Superuser mode, 2-737
- entry
 - directory, 2-207
 - link, 2-202
- ?ENUS value, 2-96
- environment, restoring the previous, 2-778
- EOT character, 2-675
- ?EPIP offset, 2-406-2-407, 2-411
- ?ERBA offset, 2-684
- ?ERCH offset, 2-684
- ?ERCS offset, 2-684
- ?ERLTH value, 2-684

ERMES file, 2-99-2-102

?ERMSG system call, 2-99-2-102

ERPFL pipe is full, 2-417

error

- codes, and text, 2-683
- message file, 2-99
- reporting, 1-9
- return, 1-2
- statistics, BSC, 2-686

error codes, 1-9

?ERSCH system call, 2-102

?ESBB value, 2-614

?ESBE value, 2-614

escape key, characteristics, and device types, 2-200

?ESCP value, 2-613

?ESCR offset, 2-606, 2-612

?ESDD value, 2-613

?ESED value, 2-613

?ESEP offset, 2-606, 2-612

?ESFC offset, 2-606, 2-612

?ESFF system call, 2-103, 2-213

?ESGT value, 2-614

?ESLN value, 2-606, 2-612

?ESNE value, 2-614

?ESNR value, 2-613

?ESPE value, 2-614

?ESRD value, 2-613

?ESRP value, 2-614

ESS, 2-284

?ESSE value, 2-613

ETB character, 2-675

?ETBB value, 2-716, 2-722

?ETER offset, 2-406-2-407, 2-411

?ETFL offset, 2-606

?ETFT offset, 2-407, 2-606

?ETLL offset, 2-606

?ETLT offset, 2-407, 2-411, 2-606

?ETMX value, 2-406-2-408

?ETSL offset, 2-606

?ETSN offset, 2-406-2-407, 2-411

?ETSP offset, 2-407, 2-605

ETX character, 2-675

?ETXB value, 2-716, 2-722

event codes

- in system log file, B-1
- special, B-1
- standard, B-1

?EXAC value, 2-781

examining

- class scheduling, 2-42
- default access control list, 2-77
- execute-protection status, 2-141-2-142
- privilege state, 2-744
- Superprocess mode, 2-735
- Superuser mode, 2-737

exception code, 1-2

exclusion bit map packet, 2-743

EXEC (CONTROL @EXEC) family of commands, 2-438

?EXEC functions

- backing up your files, 2-110
- batch processing, 2-112
- changing queuing parameters, 2-133
- dismounting a unit (extended request), 2-132
- dismounting unlabeled and labeled tapes, 2-109
- holding, unholding, canceling queue requests (AOS/VS), 2-127
- IPC print notification, 2-122
- mounting unlabeled and labeled tapes, 2-106-2-112
- obtaining
 - EXEC status information, 2-129
 - extended status information, 2-130
 - QDISPLAY information, 2-138
 - queue names, 2-136
- queuing a file entry, 2-112
- spooling output, 2-112
- submitting a job to a MOUNT queue, 2-131
- summary of, 2-104

?EXEC system call, 2-104, 2-438

EXEC utility program, 2-104

EXECUTE command, 2-255

execute-protection status, 2-141-2-142

execution

- path, task, 2-278
- program, 1-8

exiting

- from an interrupt service routine, 2-314
- from an overlay, 2-510

?EXPO system call, 2-141

extended

- characteristics, 2-190

device characteristics, 2-679
state save area, 2-284
status information about a process, 2-900
?EXTG pseudo-operation, A-17

F

F77BUILD_SYM program, A-40
?FAAB value, 2-245-2-246
?FACA value, 2-78, 2-210, 2-246, 2-520
?FACE value, 2-78, 2-210, 2-246, 2-520
?FACO value, 2-78, 2-210, 2-246, 2-520
?FACR value, 2-78, 2-210, 2-246, 2-520
factors, bias, 2-176, 2-655
?FACW value, 2-78, 2-210, 2-246, 2-520
?FAEA value, 2-169
?FAEB value, 2-245-2-246
?FAOB value, 2-245-2-246
?FARA value, 2-169
?FARB value, 2-245-2-246
fast user device, 2-149
father process, getting the PID of a, 2-79
?FAWB value, 2-245-2-246
?FBEX offset, 2-145, 2-147
?FBSTF offset, 2-145
?FCPC offset, 2-145, 2-146, 2-147
?FCPD file type, 2-61, 2-64-2-66, 2-415, 2-852
?FCPD value, 2-167
FCU, 2-600-2-601
?FDAY system call, 2-143
?FDBA offset, 2-145, 2-147
?FDBFZ offset, 2-146
?FDBL offset, 2-145, 2-147
?FDIR file type, 2-61, 2-64-2-66, 2-415, 2-852
?FDIR value, example of, 1-6-1-7
?FDLE value, 2-169
?FEDFUNC system call, 2-144-2-158
?FEOV system call, 2-148
?FEXPR value, 2-147
FF, 2-663
?FFCC file type, 2-61, 2-852

?FFLAG offset, 2-145-2-147
?FFLPT value, 2-147
?FGLT file type, 2-61, 2-852
?FIDEF system call, 2-149-2-156, 2-794, 2-803
 warnings about, 2-156
field translation, 2-616
FILCREATE.SR sample program, A-2, A-19-A-21
file
 attribute, permanent, 2-653
 block I/O, opening a, 2-210
 changing group ACL, 2-240
 closing a, 2-38, 2-183
 complete pathname of generic, 2-239
 creating a, 2-60, 2-405
 creation and management system calls (names of), 1-6
 creation options, 2-413
 deleting a, 2-86
 descriptor information, 2-780
 directory, getting status of, 2-168-2-169
 dump, 2-381
 error message, 2-99
 flushing to disk, 2-103
 generic, 2-206
 getting ACL, 2-77, 2-174, 2-245, 2-650
 getting group ACL, 2-882
 input/output system calls (names of), 2-7-2-8
 IPC, 2-211, 2-213, 2-288
 IPC, getting status of, 2-166
 opening a, 2-405
 opening for shared-access, 2-699
 other types, getting status of, 2-168
 pointer
 getting the position, 2-220
 positioning the, 2-610, 2-707
 protected, 2-519
 protected shared, 2-701
 recreating a, 2-629
 renaming a, 2-632
 shared, 2-659, 2-699
 access, opening a, 2-405
 flushing to disk, 2-103
 specifications word, 2-412
 status information, getting, 2-163
 symbol table, 2-261
file (continued)
 system log, 2-739
 truncating a, 2-259, 2-773
 unit, getting status of, 2-165
File Editor functions
 change radix, 2-144
 delete a temporary symbol, 2-147
 disassemble an instruction, 2-146

evaluate a FED string, 2-145
 examining dump file, 2-381
 insert a temporary symbol, 2-146
 interfacing to, 2-144
 open symbol table file, 2-145
 filename
 directory, 2-207
 program (.PR), returning, 2-640
 templates, 2-208
 FILESTATUS command, 2-214
 ?FINA offset, 2-146
 ?FINST value, 2-147
 ?FIPC file type, 2-61, 2-63, 2-66-2-67, 2-415,
 2-852
 ?FIPC value, 2-211
 ?FIXMT system call, 2-157-2-158
 ?FLCC file type, 2-61, 2-852
 ?FLCHN offset, 2-161, 2-173
 ?FLCR value, 2-145
 ?FLDIS value, 2-146
 ?FLDU value, 2-167
 ?FLEFS value, 2-145
 ?FLEX offset, 2-145, 2-147
 ?FLLEN value, 2-161, 2-173
 ?FLNK file type, 2-61, 2-852
 floating-point status register, 2-285
 floating-point unit, initializing the, 2-285
 ?FLOCK system call, 2-159-2-160
 ?FLOG file type, 2-741
 ?FLOST value, 2-145
 flow control, hardware, 2-197
 output, 2-197
 ?FLPID offset, 2-161, 2-173
 ?FLREV offset, 2-161, 2-173
 ?FLRSW offset, 2-161, 2-173
 ?FLSEL offset, 2-161, 2-173
 ?FLSYM value, 2-146, 2-147
 ?FLTY offset, 2-161, 2-173
 ?FLUSH system call, 2-162
 flushing
 file descriptor information, 2-780
 shared file memory pages to disk, 2-103
 shared page to disk, 2-162
 ?FMDB value, 2-169
 ?FMEFS value, 2-147
 ?FNCC file type, 2-61, 2-852
 ?FNIR offset, 2-145
 ?FOCC file type, 2-61, 2-852
 forcing, end-of-volume on labeled tape, 2-148
 form feed character, 2-663
 Format conventions, xi
 Format conventions, v
 Forms Control Utility program, 2-600-2-601
 FORTRAN 77
 language, 1-10-1-11
 operating system interface sample program
 set, A-2-A-3, A-39
 sample program set, iv, x
 ?FPIP file type, 2-61, 2-852
 ?FPRG file type, 2-61, 2-852
 ?FPRM value, 2-169
 ?FPRV file type, 2-61, 2-415, 2-852
 ?FQUE file type, 2-61, 2-852
 frame information, stack, 2-807
 frame pointer, 1-2
 ?FRCR value, 2-144
 ?FRDIS value, 2-144, 2-146
 ?FRDTS value, 2-144, 2-147
 ?FREFS value, 2-144, 2-145, 2-147
 frequency of the system clock, 2-201
 getting the, 2-201
 ?FRESA offset, 2-145, 2-147
 ?FRESS offset, 2-145
 ?FRFNC offset, 2-145-2-147
 ?FRITS value, 2-144, 2-146
 ?FROST value, 2-144-2-145
 ?FRRR offset, 2-146
 ?FSDF file type, 2-61, 2-852
 ?FSHB value, 2-169
 ?FSNL offset, 2-146, 2-147
 ?FSNM offset, 2-146, 2-147
 ?FSPR file type, 2-61, 2-63, 2-66-2-67, 2-852
 ?FSTAT system call, 2-163-2-164, 2-214
 ?FSTF file type, 2-61, 2-852
 ?FSVAL offset, 2-146, 2-147
 ?FSVLL offset, 2-146, 2-147
 ?FTCK value, 2-161

?FTER value, 2-161
 ?FTEX value, 2-161
 ?FTOD system call, 2-171
 ?FTPN value, 2-161
 ?FTSH value, 2-161
 ?FTXT file type, 2-61, 2-415, 2-852
 ?FUDA value, 2-169
 ?FUDF file type, 2-61, 2-410, 2-415, 2-852
 ?FULA value, 2-173
 full process name, 2-265, 2-521
 ?FUNLOCK system call, 2-172-2-173
 ?FUNX file type, 2-61, 2-852
 ?FUPF file type, 2-61, 2-852
 ?FWFI offset, 2-146
 ?FWFL value, 2-161

G

?GACL system call, 2-174
 ?GARG value, 2-252, 2-254
 GATE.ARRAY.SR sample program, A-2, A-17
 ?GBIAS system call, 2-176
 ?GCFC value, 2-255
 ?GCHR system call, 2-335
 See also The ?GECHR system call
 ?GCLOSE system call, 2-183-2-184, 2-214, 2-216, 2-412
 ?GCMD value, 2-252, 2-254, 2-255
 ?GCNT value, 2-252, 2-254
 ?GCPCN offset, 2-71, 2-87, 2-175, 2-632.2, 2-652, 2-654
 ?GCPFW offset, 2-71, 2-87, 2-175, 2-632.2, 2-652, 2-654
 ?GCPLT value, 2-71, 2-87, 2-175, 2-632.2, 2-652, 2-654
 ?GCPN system call, 2-185
 ?GCRB system call, 2-186
 ?GDAY system call, 2-187
 ?GDLC value, 2-253-2-255
 ?GDLM system call, 2-188-2-189
 ?GECHR system call, 2-190-2-192
 generic file, 2-206
 complete pathname of, 2-239

get/set class ID code, 2-36-2-56
 ?GFCF value, 2-252
 ?GHRZ system call, 2-201
 ?GLINK system call, 2-202
 ?GLIST system call, 2-203
 global port number
 and PID association, 2-219
 local port number with, 2-308
 modifying a ring field within a, 2-289
 returning a, 2-288
 ring field with, 2-308
 translate local to global equivalent, 2-770
 ?GMEM system call, 2-204
 ?GMES value, 2-252, 2-254, 2-255
 GMT, 2-247, 2-400
 ?GNAME system call, 2-205, 2-239
 ?GNFN system call, 1-1, 2-207-2-209
 example of, A-26
 ?GNUM offset, 2-251, 2-253, 2-255
 ?GOPEN system call, 2-168, 2-210-2-216
 example of, A-26
 ?GPID system call, 2-217
 ?GPORT system call, 2-219
 ?GPOS system call, 2-220-2-221
 ?GPRNM system call, 2-222
 ?GRAPH_CLOSE_PIXELMAP function, 2-224, 2-229
 ?GRAPH_CRE... offsets and values, 2-228
 ?GRAPH_CREATE_MEMORY_PIXELMAP function, 2-224-2-225, 2-227
 ?GRAPH_GET_DRAW_ORIGIN function, 2-224
 ?GRAPH_MAP... offsets and values, 2-233
 ?GRAPH_MAP_PIXELMAP function, 2-224, 2-233
 ?GRAPH_OPEN... offsets and values, 2-226-2-258
 ?GRAPH_OPEN_WINDOW_PIXELMAP function, 2-224, 2-226
 ?GRAPH_PIXELMAP_STATUS function, 2-224, 2-229
 ?GRAPH_PIXSTAT... offsets and values, 2-229-2-258
 ?GRAPH_PKT... offsets and values, 2-223, 2-225, 2-228-2-229, 2-234
 ?GRAPH_RDPAL... offsets and values, 2-236
 ?GRAPH_READ_PALETTE function, 2-224, 2-235-2-236

?GRAPH_RECT_STATE_DISABLE value, 2-230, **2-232**

?GRAPH_RECT_STATE_ENABLE value, 2-230, **2-232**

?GRAPH_SET_CLIP... offsets and values, 2-231-2-258

?GRAPH_SET_CLIP_RECTANGLE function, 2-224, 2-230

?GRAPH_SET_DRAW_ORIGIN function, 2-224

?GRAPH_UNMAP_PIXELMAP function, 2-224, 2-232, 2-234

?GRAPH_WRITE_PALETTE function, 2-224, **2-234**

?GRAPH_WRPAL... offsets and values, 2-235

graphics channel, output, 2-225

?GRAPHICS functions

- closing a pixel map, 2-229
- creating a pixel map in memory, 2-227
- getting the coordinates of the draw origin, 2-237-2-238
- getting the status of a pixel map, 2-229
- mapping a pixel map into a program's address space, 2-233
- opening a graphics window's pixel map, 2-225-2-226
- reading from a palette, 2-235-2-236
- setting the clip rectangle, 2-230
- setting the draw origin, 2-236
- transferring data between pixel maps and files, 2-232-2-233
- unmapping a pixel map from a program's address space, 2-234-2-235
- writing to a palette, 2-234

graphics output channel, 2-225

?GRAPHICS system call, 2-191, **2-223-2-238**

- example of, A-58

graphics window, 2-226

?GRAPHICS_GET_DRAW_ORIGIN function, 2-237-2-238

?GRAPHICS_GET_ORG... offsets and values, 2-238

GRAPHICS_SAMPLE.SR sample program, A-2, **A-56-A-60**

?GRAPHICS_SET_DRAW_ORIGIN function, 2-236

?GRAPHICS_SET_ORG... offsets and values, 2-237

?GRCH offset, 2-649

Greenwich Mean Time, 2-400

?GREQ offset, 2-251, 2-252

?GRES offset, 2-251, 2-252, 2-254, 2-255

?GRIH offset, 2-649

?GRLTH value, 2-649

?GRNAME system call, 2-239

group

- access control list, 2-548, 2-883-2-885
- buffer, 2-241-2-242
- list, 2-884
- name defined, 2-548

?GROUP system call, 2-240, 2-548

?GROUP_... offsets and values, 2-241-2-242

GRP MASM macro, 2-101

?GRPH offset, 2-649

?GRRH offset, 2-649

?GSHPT system call, 2-243

?GSID system call, 2-244

?GSW offset, 2-251, 2-252, 2-255

?GSWS value, 2-253-2-255

?GTACP system call, 2-245-2-246

?GTIME system call, 2-247-2-249

?GTMES system call, **2-250**, 2-543

- examples of, A-26, A-33, A-37

?GTNAM system call, 2-256-2-257

?GTOD system call, 2-258

?GTRUNCATE system call, 2-213, **2-259**

?GTSVL system call, 2-261-2-262

?GTSW value, 2-252, 2-253-2-255

?GUHBP offset, 2-264

?GUHFL offset, 2-264

?GUHFN offset, 2-264

?GUHID offset, 2-264

?GUHLN offset, 2-264

?GUHLR offset, 2-264

?GUHP0 value, 2-264

?GUHPI system call, 2-263

?GUID value, 2-264

?GUNM system call, 2-265

?GVPID system call, 2-266

H

handler service routine

- device interrupt, 2-275

- fast device interrupt, 2-156
- ?HAPH array offset, 2-287, 2-812
- ?HAPL array offset, 2-287, 2-812
- ?HARAY array offset, 2-287, 2-812
- hardware processor identification, unique, 2-263
- HEAR.SR sample program, A-1, **A-3-A-4**
- hertz, definition of, 2-645
- ?HIBUF offset, 2-287
- ?HIEND offset, 2-287
- high-level language interface, 1-1, 1-10-1-11
- high-level language sample program set, iv, x
- high-order bits, 1-5
- ?HIST offset, 2-287
- histogram
 - killing a, 2-329
 - multiprocessor, 2-393
 - starting a, 2-286, 2-393, 2-810
 - uniprocessor, 2-393
- ?HIWDS offset, 2-287
- ?HNAME system call, 2-267-2-268
- /HOFC switch, 2-180
- host
 - ID, 2-168, 2-217, 2-267
 - local, 2-28
 - remote, 2-28, 2-639
- hostname, 2-267
- hotspot, cursor, 2-573
- ?HPRH array offset, 2-287, 2-812
- ?HPRL array offset, 2-287, 2-812
- ?HRDFLC hardware flow control offset, 2-197
- ?HRDFLC value, 2-180
- ?HSBH array offset, 2-287, 2-812
- ?HSBL array offset, 2-287, 2-812
- ?HSIH array offset, 2-287, 2-812
- ?HSIL array offset, 2-287, 2-812
- ?HTTH array offset, 2-287, 2-812
- ?HTTL array offset, 2-287, 2-812
- ?HWBUF offset, 2-811
- ?HWEND offset, 2-811
- ?HWLTH value, 2-287, 2-811
- ?HWST offset, 2-811

- ?HWWDS offset, 2-811

I

- I/O
 - and MCA protocol, 2-527
 - physical block, 2-525, 2-592
- I/O
 - and new file system, 2-210
 - block, 2-19, 2-596
 - conditional, 2-220, 2-707
 - disk block, 2-598
 - file, 2-220, 2-707
 - MCA block, 2-599
 - modified sector, 2-216
 - physical block, 2-210
 - reading and writing record, 2-604
 - tape block, 2-599
 - writing block or record, 2-844
- ?IBAD offset, 2-39, 2-221, 2-407, 2-410, 2-605, 2-609
- ?IBIN value, 2-408, 2-608, 2-611
- ?IBLT value, 2-406, 2-407, 2-606, 2-608
- ?ICH offset, 2-39, 2-221, 2-406-2-408, 2-605-2-606
- ?ICRF value, 2-408, 2-412, **2-608**
- ID
 - host, 2-168, 2-217, 2-267
 - pixel map, 2-226
- ?ID8 value, 2-410, 2-417
- ?ID16 value, 2-410, 2-417
- ?ID5 value, 2-410, 2-417
- ?ID6 value, 2-410, 2-417
- ?ID62 value, 2-410, 2-417
- ?ID7 value, 2-417
- ?IDAM value, 2-410, 2-417
- ?IDEF system call, 2-269-2-272
- ?IDEL offset, 2-39, 2-221, 2-406, 2-407, 2-411, 2-416, 2-605, 2-608, 2-610
- identification, unique processor hardware, 2-263
- identification bloc, disk, 2-291
- identifier
 - checking volume, 2-35
 - host, 2-267
 - system, 2-244, 2-719
 - unique task, 2-688, 2-690, 2-777
- ?IDGOTO system call, 2-278

?IDKIL system call, 2-279
 example of, A-29

?IDPH offset, 2-306, 2-310, 2-311

?IDPN offset, 2-296

?IDPRI system call, 2-280

?IDRDY system call, 2-281

?IDSTAT system call, 2-282

?IDSUS system call, 2-283

?IESS system call, 2-284

?IEXO value, 2-408, 2-412

?IFNBK value, 2-296, 2-311

?IFNP offset, 2-39, 2-221, 2-407, 2-411, 2-605, 2-610

?IFNSP value, 2-306, 2-311

?IFOP value, 2-609

?IFPR value, 2-296, 2-311

?IFPU system call, 2-285

?IFRFM value, 2-296, 2-311

?IFRING value, 2-296, 2-311

?IFSOV value, 2-296, 2-311

?IFSTM value, 2-306, 2-311

?IHIST system call, 2-286

?IIPC value, 2-409, 2-611

?ILKUP system call, 2-288
 examples of, A-4, A-7

?ILTH offset, 2-296, 2-306, 2-310, 2-311

?IMERGE system call, 2-289

?IMFF value, 2-409

?IMHN value, 2-609

?IMIO offset, 2-527

?IMIO value, 2-598

?IMNH value, 2-409

?IMP2 value, 2-406, 2-409, 2-609

implicit system call, A-39

?IMRS offset, 2-39, 2-68, 2-221, 2-407, 2-410, 2-605, 2-609
 for pipe size, 2-416

?IMSG system call, 2-290

index levels, 2-64

indicating, prior rescheduling state, 2-88

?INID value, 2-436

?INIT system call, 2-291, 2-887

initial IPC message, 2-250

INITIALIZE CLI command, 2-292

initializing
 extended state save area, 2-284
 floating-point unit, 2-285
 job processor, 2-317
 logical disk, 2-291, 2-630
 logical disk (extended), 2-886

initiating, a task, 2-747

initiation queue, task, 2-294

INRING.SR sample program, A-2, A-16-A-18

intelligent
 asynchronous controller, 2-17, 2-41, 2-271, 2-405, 2-534, 2-604
 controller, 2-190

?INTEO value, 2-411

interface
 assembly language, 1-5-1-9
 high-level language, 1-1
 operator, 2-424

internal time, returning the OS-format, 2-313

interprocess communications system calls
 (names of), 2-10

interprocess signaling mechanism, 2-688, 2-690, 2-691, 2-845

interrupt
 control-character terminal, 2-333, 2-334
 disabling terminal, 2-403
 enabling terminal, 2-404
 handler service routine
 device, 2-275
 fast device, 2-156
 sequences, keyboard, 2-332
 service message, 2-290
 service routine, 2-151, 2-271
 exiting from an, 2-314
 transmitting a message from an, 2-157, 2-315
 task, 2-278
 terminal, 2-293, 2-335

intertask message
 receiving an, 2-627
 receiving without waiting, 2-628
 transmitting an, 2-898, 2-899

?INTWT system call, 2-293

IOC device, 2-151, 2-271

?IOPH offset, 2-296

?IOPN offset, 2-306, 2-310, 2-311

?IOSZ value, 2-39, 2-221, 2-406, 2-407, 2-606, 2-608

IPC
 file, 2-211, 2-213, 2-288
 getting status of, 2-166

message, 2-219, 2-289
 CLI-format, 2-307
 receiving an, 2-295
 sending an, 2-305
 sending and then receiving an, 2-309
 sending via ?PROC, 2-543
 ?IPKL value, 2-406, 2-408, 2-608
 ?IPLTH value, 2-296, 2-306
 ?IPRLTH value, 2-310
 ?IPST value, 2-608, 2-610, 2-707-2-708
 ?IPTR offset, 2-296, 2-300, 2-306, 2-310,
 2-311
 ?IQTSK system call, 2-294
 ?IRCL offset, 2-39, 2-221, 2-407, 2-411, 2-605,
 2-609, 2-707-2-708
 ?IREC system call, 1-2, **2-295-2-297**
 example of, A-4
 ?IRES offset, 2-39, 2-221, 2-407, 2-410, 2-605,
 2-609
 ?IRLR offset, 2-39, 2-221, 2-407, 2-411, 2-605,
 2-610
 ?IRLT offset, 2-310, 2-311
 ?IRMV system call, 2-151, 2-276, **2-304.9**
 ?IRNH offset, 2-39, 2-221, 2-407, 2-411,
 2-605, 2-610, 2-707-2-708
 ?IRNW offset, 2-39, 2-221, 2-407, 2-411,
 2-605, 2-610
 ?IRPT offset, 2-310, 2-311
 ?IRSV offset, 2-310, 2-311
 ?IS.R system call, 2-309-2-312
 ISC device, 2-271
 ?ISEND system call, 2-305-2-306
 example of, A-7
 ?ISFL offset, 2-296, 2-306, 2-310, 2-311,
 2-543
 ?ISPLIT system call, 2-297, **2-308**

?ISTI offset, 2-39, 2-221, 2-406-2-410, 2-412,
 2-413, 2-605, 2-608, 2-609, 2-707
 ?ISTO offset, 2-39, 2-221, 2-406, 2-407, 2-409,
 2-410, 2-605, 2-609
 ISYS FORTRAN 77 function, 1-10
 ITB character, 2-676
 ?ITIME system call, 2-313
 ?IUFL offset, 2-73, 2-296, 2-297-2-304.8,
 2-306, 2-310, 2-311, 2-543
 example of, 2-297
 ?IXIT system call, 2-151, 2-276, 2-277, **2-314**
 ?IXMT system call, 1-2, 2-151, **2-315**

J

job processor
 getting the status of a, 2-324
 initializing a, 2-317
 moving to a new logical processor, 2-320
 releasing a, 2-322
 ?JPI_PKT... offsets and values, 2-318-2-319
 ?JPID_MAX value, 2-319, 2-321, 2-323
 ?JPID_MIN value, 2-319, 2-321, 2-323
 ?JPINIT system call, 2-317-2-336
 JPLCS instruction, 2-319
 ?JPM_PKT... offsets and values, 2-321
 ?JPMOV system call, 2-320
 ?JPR_PKT... offsets and values, 2-323
 ?JPREL system call, 2-322-2-323
 ?JPS_GEN value, 2-325
 ?JPS_GEN... offsets and values, 2-326
 ?JPS_PKT... offsets and values, 2-325
 ?JPS_SPEC value, 2-325
 ?JPS_SPEC... offsets and values, 2-327
 ?JPSTAT system call, 2-324
 JPSTATUS instruction, 2-324

K

- Kanji character sets, 2-192
 - and VT100, 2-192
 - Japanese, 2-192
 - Taiwanese, 2-192
- ?KCALL system call, 2-328
- keyboard interrupt sequences, 2-332
- ?KHIST system call, 2-329
- ?KILAD system call, 2-330
- kill-processing routine, 2-279, 2-330
- ?KILL system call, 2-331
 - example of, A-33
- killing
 - histogram, 2-329
 - task, 2-279, 2-331, 2-510
 - tasks of a specified priority, 2-533
- ?KINTR system call, 2-332
- ?KIOFF system call, 2-333
- ?KION system call, 2-334
- ?KWAIT system call, 2-335

L

- ?LABEL system call, 2-336
- labeled
 - diskette, 2-336
 - magnetic tape, 2-35, 2-38, 2-336, 2-418
 - forcing end-of-volume, 2-148
 - trailer, 2-38
- LAC device, 2-41
- language
 - assembly, 1-1
 - interface, high-level, 1-1, 1-10-1-11
- Language Front-end Processor, options, **2-199**
- ?LB8 value, 2-337, 2-338
- ?LB16 value, 2-338
- ?LB5 value, 2-338
- ?LB6 value, 2-338
- ?LB62 value, 2-338
- ?LB7 value, 2-338
- ?LBAC offset, 2-337-2-338
- ?LBAM value, 2-338
- ?LBDV offset, 2-337-2-338
- ?LBFG offset, 2-337-2-338
- ?LBIM value, 2-338
- ?LBLN value, 2-337
- ?LBMF value, 2-338
- ?LBMP value, 2-338
- ?LBMR value, 2-338
- ?LBMS value, 2-338
- ?LBOI offset, 2-337-2-338
- ?LBSC value, 2-338
- ?LBST offset, 2-337-2-338
- ?LBUV offset, 2-337-2-338
- ?LBVD offset, 2-337-2-338
- LCALL instruction, A-14
- ?LDMA event code, 2-361
- LDU images
 - initializing, 2-340
 - mirroring and synchronizing, 2-385
- ?LDU_IMAGE_HARDWARE_MIRRORED value, 2-345
- ?LDU_IMAGE_REMOVED value, 2-344
- ?LDU_MIRROR_BEING_SYNCHRONIZED value, 2-344
- ?LDU_MIRRORED value, 2-344
- ?LDU_PKT... offsets and values, 2-343-2-347
- ?LDU_PRIMARY_IMAGE value, 2-345
- ?LDUINFO system call, 2-340-2-350
- ?LDUINFO... offsets and values, 2-341, 2-345-2-346
- least significant bit, 1-5
- leaving
 - privilege state, 2-744
 - Superprocess mode, 2-735
 - Superuser mode, 2-737
- LEF mode, 2-81, 2-83, 2-350
 - disabling, 2-350
 - enabling, 2-351
 - status, returning, 2-352
- ?LEFD system call, 2-350
- ?LEFE system call, 2-351
- ?LEFS system call, 2-352
- ?LFOP value, 2-357
- line
 - break, 2-40
 - BSC
 - disabling a, 2-661
 - receiving information over a, 2-709
 - sending information over a, 2-720
 - printer, data channel, 2-70, 2-600-2-602

- link entry, 2–202
- Link utility program, 1–8
- listing
 - directory entries, 2–207
 - shared partition size, 2–243
 - unshared memory parameters, 2–383
- ?LMAP system call, 2–353
- ?LMAX event code, 2–361
- LOAD CLI command, 2–110
- LOAD_II CLI command, 2–110
- loading
 - overlay, 2–511
 - program file, 2–637
- ?LOC_... offsets and values, 2–355–2–356
- local
 - host, process or queue name, 2–28
 - port number, 2–219, 2–770
- locality
 - process, 2–514
 - scheduling matrix, class, 2–51
 - user, changing, 2–354
- ?LOCALITY system call, 2–354
- locating, process name, 2–28
- locking, an object, 2–159
- log file
 - ?GROUP entry, 2–241
 - system, 2–360, 2–739
 - system call, 2–357
- ?LOGCALLS system call, 2–357–2–359
- LOGCALLS utility program, 2–358
- ?LOGDREC value, 2–359
- ?LOGEV system call, 2–360
- ?LOGF16U value, 2–358
- logging, system calls, 2–357
- ?LOGHREC value, 2–358
- logical
 - address, 2–781
 - address space, mapping a device into, 2–373
 - disk
 - information, returning, 2–340
 - initialized, 2–340, 2–630
 - initializing a, 2–291
 - initializing a (extended), 2–886
 - unit image, synchronized, 2–887
 - processor
 - class assignments, 2–362
 - creating a, 2–365
 - deleting a, 2–367
 - getting the status of a, 2–369
 - moving a job processor to, 2–320
 - shared memory, 2–243
- low-order bits, 1–5
- lower ring
 - loading and stopping, 2–642
 - mapping, 2–353
- ?LPC_PKT... offsets and values, 2–366
- ?LPCL_PKT... offsets and values, 2–363–2–364
- ?LPCLASS system call, 2–362–2–364
- ?LPCREA system call, 2–365–2–366
- ?LPD_PKT... offsets and values, 2–368
- ?LPDELE system call, 2–367–2–368
- ?LPID_MAX value, 2–319, 2–321, 2–365
- ?LPID_MIN value, 2–319, 2–321, 2–365
- ?LPS_FUNC_MAX value, 2–370
- ?LPS_FUNC_MIN value, 2–370
- ?LPS_GEN... offsets and values, 2–370
- ?LPS_PKT... offsets and values, 2–370
- ?LPS_SPEC... offsets and values, 2–370–2–372
- ?LPSTAT system call, 2–369–2–370
- ?LSMI event code, 2–361, 2–742
- ?LSTART value, 2–357
- ?LTSF event code, 2–742
- ?LUMAX value, 2–742
- ?LUMI event code, 2–361

M

- Macroassembler program, iv, 1–8, x
- magnetic tape, 2–417
 - data compression, 2–215
 - densities
 - absolute, 2–212
 - relative, 2–212
 - labeled, 2–35, 2–336, 2–418
 - unit, Model 6352, 2–212, 2–214, 2–338, **2–412**
- magnetic tape drives, and native mode, 2–215
- maintaining, and creating an operator interface, 2–424
- manager, queued task, 2–294
- manipulating
 - pixel maps, 2–223
 - the system log file, 2–739
 - windows, 2–813
- map
 - data channel, 2–729

definition table, 2-153-2-154, 2-272, 2-731
 pixel, 2-223
 ?MAPDV system call, 2-373-2-375
 ?MAPDV_PKT_PKTID value, 2-374, 2-376
 mapping
 device into logical address space, 2-373
 lower ring, 2-353
 mask, bit, 1-4
 MASM, 1-5-1-9
 MASM.PR program, 1-8
 MASM.PS file, 1-8
 MASM_32CHAR.PS file, 1-9, A-54, A-74
 ?MAXIMAGES value, 2-895
 ?MBAH offset, 2-378, 2-380
 ?MBBC offset, 2-378, 2-380
 ?MBCH offset, 2-378, 2-380
 ?MBFC system call, 2-377-2-378
 ?MBID offset, 2-378, 2-380
 ?MBLTH value, 2-378, 2-380
 ?MBNHR value, 2-390
 ?MBNLD value, 2-390, 2-391
 ?MBOOP value, 2-390
 ?MBTC system call, 2-379-2-380
 ?MBTRP value, 2-390
 ?MBWAIT value, 2-390
 MCA
 block I/O, 2-599
 unit, 2-210
 MCA protocol, with I/O, 2-527
 ?MCOBIT value, 2-56
 MCP1 device, 2-41
 ?MCPID value, 2-56
 ?MCRNG value, 2-56
 ?MDAC offset, **2-375**, 2-376
 ?MDAL offset, **2-375**, 2-376
 ?MDCL offset, 2-374, **2-375**, 2-376
 ?MDDL offset, 2-374, **2-375**, 2-376
 ?MDDT offset, 2-374, **2-375**, 2-376
 ?MDID offset, **2-375**, 2-376
 ?MDIL offset, 2-374, **2-375**, 2-376
 ?MDLA offset, 2-374, **2-375**, 2-376
 ?MDLL offset, **2-375**, 2-376
 ?MDN0—?MDN1 offsets, 2-374-2-376
 ?MDNL offset, **2-375**, 2-376
 ?MDNP offset, **2-375**, 2-376
 ?MDOP offset, 2-374, **2-375**, 2-376
 ?MDOX offset, **2-375**, 2-376
 ?MDP0 value, 2-375
 ?MDPC offset, 2-374, **2-375**, 2-376
 ?MDPK offset, 2-374, **2-375**, 2-376
 ?MDPL offset, 2-374, **2-375**, 2-376
 ?MDPV value, 2-373
 ?MDRE offset, **2-375**, 2-376
 ?MDRL offset, 2-374, **2-375**, 2-376
 ?MDRP offset, 2-374, **2-375**, 2-376
 ?MDRT offset, 2-374, **2-375**, 2-376
 ?MDUMP system call, **2-381-2-382**, 2-471
 ?MEM system call, 2-383
 ?MEMI system call, 2-384
 memory
 address, 1-1
 dump, 2-471
 image, dumping a, 2-381
 logical shared, 2-243
 management system calls (names of), 2-3
 mapped device, 2-373-2-374
 pages
 changing (unshared), 2-384
 flushing (shared file) to disk, 2-103
 undedicated, 2-204
 parameters (unshared), listing, 2-383
 read/write access to, 2-782
 meridian, prime, 2-247, 2-400
 message
 CLI, 2-250
 error file, 2-99
 initial IPC, 2-250
 IPC, 2-219, 2-295, 2-305
 task, 2-771
 terminal, 2-681
 ?MFBRK value, 2-390
 ?MFSYM value, 2-390
 microcode for a job processor, 2-318
 ?MIFUN offset, 2-389-2-390
 ?MII1 and ?MII2 values, 2-390
 ?MIID offset, **2-389**, 2-391
 ?MILD offset, **2-389**, 2-391
 ?MIOP offset, 2-389-2-390
 ?MIPHI offset, 2-389-2-390

?MIPLO offset, 2-389-2-390
 ?MIPUL offset, **2-389**, 2-391
 ?MIR1-?MIR4 offsets, 2-389-2-391
 ?MIRES offset, 2-389-2-390
 ?MIRROR system call, 2-385-2-389
 ?MIRROR_... offsets and values, 2-386-2-388
 mirroring, LDU images, 2-385, 2-886-2-897
 ?MMAP value, 2-374, 2-376
 mode, binary, 2-40
 Model 6236-6240 disks, 2-25
 Model 6352 magnetic tape unit, 2-212, 2-214, 2-338, **2-412**
 modem
 carrier detect, 2-198
 connection, timing, 2-199
 hardware input flow control, 2-198
 options, **2-198**
 user access, 2-198
 modem support, 2-180
 modified sector I/O, 2-20, 2-216
 modifying, ring field within a global port number, 2-289
 month, current, 2-187
 most significant bit, 1-5
 MOUNT command, 2-210
 mouse movement, 2-191
 moving
 bytes from a customer buffer, 2-377
 bytes to a customer buffer, 2-379
 job processor to a new logical processor, 2-320
 ?MPH_... offsets and values, 2-393-2-398
 ?MPHIST system call, 2-393-2-395
 ?MPHIST_... offsets, 2-396-2-398
 MRC device routes
 current, 2-58
 diverted, 2-58
 primary, 2-58
 secondary, 2-58
 ?MRDO value, 2-374, 2-376
 multipoint control station, 2-715
 multipoint tributary station, 2-716
 multiprocessor histogram, 2-393
 multiprocessor management system calls (names of), 2-11, 2-13

multitask scheduling, enabling, 2-102
 multitasking system calls (names of), 2-9-2-10
 ?MXFN value, 2-345, 2-895
 ?MXHN value, 2-267
 ?MXLPN value, 2-63
 ?MXPL value, 2-202, 2-203, 2-641, 2-694, 2-695
 ?MXUN value, 2-265, 2-908
 ?MYTID system call, 2-399
 example of, A-29

N

NAK character, 2-676
 name, full process, 2-265, 2-521
 National Bureau of Standards, 2-590
 NBS, 2-590
 new file system, 2-205
 New Line character, 2-663
 NEWTASK.SR sample program, A-2, **A-29-A-31**
 ?NFKY offset, 2-208
 ?NFLN value, 2-208
 ?NFSNM offset, 2-208
 ?NFRS offset, 2-208
 ?NF'TP offset, 2-208
 NL, 2-663
 normal return, 1-2
 ?NPAL offset, 2-218
 ?NPAP offset, 2-218
 ?NPFW offset, 2-218
 ?NPKEY offset, 2-218
 ?NPLTH value, 2-218
 ?NP'NEN offset, 2-218
 ?NPNUM offset, 2-218
 ?NPPR offset, 2-218
 ?NPRS1 offset, 2-218
 ?NTIME system call, 2-400
 ?NTRN value, 2-722
 null character, 1-10, 2-663
 number
 channel, 2-226, 2-569
 global port, 2-219, 2-288, 2-289, 2-770
 local port, 2-219, 2-770
 window ID, 2-226, 2-569

number specification
decimal, v, xi
octal, v, xi

O

?OBBQ value, 2-426
?OBCD value, 2-426
?OBCE value, 2-426
?OBGM value, 2-426
?OBHD value, 2-426
obituary message, 2-298
object
locking an, 2-159
unlocking an, 2-172
?OBLD value, 2-426
?OBLT value, 2-426
?OBMI value, 2-426
?OBPR value, 2-426
?OBQU value, 2-426
?OBUD value, 2-426
?OBUT value, 2-426
?OCIL offset, 2-432-2-433
?OCOL offset, 2-432-2-433
?OCR1-?OCR9 offsets, 2-432-2-433
?OCRD offset, 2-432-2-433
octal number specification, v, xi
?ODBS value, 2-215
?ODBY offset, 2-215, 2-216
?ODF1 offset, 2-211, 2-215, 2-216
?ODHD offset, 2-215, 2-216
?ODHS value, 2-215
?ODIS system call, 2-403
?ODMB value, 2-215
?ODND value, 2-215
?ODP0 value, 2-215
?ODSEC offset, 2-215, 2-216
?ODST value, 2-215
?ODTEO value, 2-215
?ODTL value, 2-210, 2-211, 2-213, **2-214**,
2-215, 2-259
?ODTP value, 2-215
?ODTRK offset, 2-215, 2-216
?OEBL system call, 2-404
?OFCE value, 2-409, 2-410, 2-413
?OFCE value, 2-409, 2-410, 2-413
?OFE2 offset, 2-428-2-429
?OFEI offset, 2-428-2-429
?OFEO offset, 2-428-2-429
?OFER offset, 2-428-2-429
offset, 1-3
?OFID value, 2-429, 2-430
?OFIN value, 2-409, 2-414
?OFIO value, 2-409, 2-414
?OFOT value, 2-409, 2-414
?OIGB value, 2-436
?OIGN value, 2-436
?OIIL offset, 2-435-2-436
?OIN2 offset, 2-435-2-436
?OIND offset, 2-435-2-436
?OINL value, 2-435
?OINP offset, 2-435-2-436
?OINR offset, 2-435-2-436
?OINT offset, 2-435-2-436
?OIOL offset, 2-435-2-436
old file system, 2-205
?OMBFM value, 2-410
?OMSTR value, 2-410
?ONE2 offset, 2-427
?ONEI offset, **2-427**, 2-428
?ONEO offset, **2-427**, 2-428
?ONER offset, 2-427
?ONID value, 2-427
?OOF2 and ?OOF3 offsets, 2-428-2-429
?OOFE offset, 2-428-2-429
?OOFL value, 2-428
?OOFN offset, 2-428-2-429
?OOFT offset, 2-428-2-429
?OOG2 offset, 2-427
?OOGT offset, 2-427
?OON2 and ?OON3 offsets, 2-427
?OONE offset, 2-427
?OONL value, 2-427

- ?OONN offset, 2-427
- ?OONT offset, 2-427
- ?OPAM value, 2-35, 2-212-2-214
- ?OPCH offset, 2-211, 2-212
- ?OPD0-?OPD2 values, 2-108
- ?OPDH value, 2-35, 2-212, 2-213
- ?OPDL value, 2-35, 2-212, 2-213
- ?OPDM value, 2-35, 2-212, 2-213
- ?OPEH offset, 2-213
- ?OPEN system call, 1-3, 2-160, **2-405**
 - examples of, A-3, A-11, A-14, A-16, A-19, A-22, A-26, A-29, A-33, A-58
- opening
 - file, 2-405
 - file for block I/O, 2-210
 - file for shared access, 2-699
 - protected shared file, 2-701
- ?OPER functions
 - ?OPINFO, 2-425, 2-435-2-436
 - ?OPOFF, 2-425, 2-428-2-429
 - ?OPON, 2-425, 2-427
 - ?OPRCV, 2-425, 2-432-2-433
 - ?OPRESP, 2-425, 2-434
 - ?OPSEND, 2-425, 2-432
- ?OPER system call, **2-424-2-425**, 2-438
- operating system, getting information, 2-692-2-692a
- operator
 - !, B, and S, 1-4
 - interface, 2-424
 - process/current process communication, 2-437
- ?OPEW offset, 2-213
- ?OPEX commands, 2-476
 - access, 2-442
 - align, 2-443
 - allocate, 2-443
 - batch_list, 2-444
 - batch_output, 2-445
 - binary, 2-446
 - brief, 2-447
 - cancel, 2-448
 - close, 2-448
 - consolestatus, 2-449-2-450
 - continue, 2-451
 - CPL, 2-452
 - create, 2-453
 - defaultforms, 2-454
 - delete, 2-454
 - disable, 2-455-2-456
 - dismounted, 2-456
 - elongate, 2-457
 - enable, 2-458-2-459
 - even, 2-460
 - flush, 2-461
 - font, 2-461
 - forms, 2-462
 - halt, 2-463
 - headers, 2-464
 - hold, 2-465
 - limit, 2-466
 - logging, 2-467
 - lpp, 2-469
 - mapper, 2-470
 - mdump, 2-471
 - message, 2-471
 - modify, 2-471
 - mounted, 2-472
 - mountstatus, 2-473
 - operator, 2-476
 - pause, 2-477
 - premount, 2-478
 - priority, 2-479-2-480
 - prompts, 2-480
 - purge, 2-480
 - qpriority, 2-481-2-482
 - refused, 2-483
 - release, 2-484
 - restart, 2-485
 - silence, 2-486
 - spoolstatus, 2-487-2-490
 - stack, 2-490-2-492
 - start, 2-492-2-494
 - status, 2-494-2-498
 - stop, 2-499
 - terminate, 2-499
 - trailers, 2-500
 - unhold, 2-501
 - unitstatus, 2-502-2-504
 - unlimit, 2-504
 - unsilence, 2-505
 - verbose, 2-508
 - xbias, 2-508
- ?OPEX system call, 2-437-2-472
- ?OPFC offset, 2-168, 2-213
- ?OPFL offset, 2-212, 2-213, 2-214
- ?OPIL offset, 2-434
- ?OPINFO function, 2-425
- ?OPK2 offset, 2-425-2-426
- ?OPKT offset, 2-425-2-426
- ?OPLT value, 2-211, 2-213
- ?OPMBF value, 2-212
- ?OPMD value, 2-212
- ?OPME value, 2-212, 2-213

?OPMST value, 2-212
 ?OPNL value, 2-425
 ?OPOFF function, 2-425
 ?OPON function, 2-425
 ?OPPH offset, 2-211
 ?OPRCV function, 2-425
 ?OPRESP function, 2-425
 ?OPSEND function, 2-425
 ?OPSP offset, 2-425-2-426
 ?OPTY offset, 2-211, 2-213
 ?OPXL value, 2-215, 2-216
 ?OPXP bit, 2-214
 ?OPXP value, 2-212
 ?OPXS offset, 2-215, 2-216
 ?ORC2-?ORC4 offsets, 2-432-2-433
 ?ORCL value, 2-432
 ?ORCN offset, 2-432-2-433
 ?ORCP offset, 2-432-2-433
 ?ORCQ offset, 2-432-2-433
 ?ORCS offset, 2-432-2-433
 ?ORCT offset, 2-432-2-433
 ?ORDS value, 2-66-2-67
 ?ORDY value, 2-66-2-67
 ?ORE2 offset, 2-425-2-426
 ?ORES offset, 2-425-2-426
 ?OREV offset, 2-425-2-426
 ?ORFX value, 2-66-2-67
 ?ORLC value, 2-431
 ?ORLO value, 2-431
 ?ORMNV value, 2-431
 ?ORMU value, 2-431
 ?ORP2-?ORP4 offsets, 2-434
 ?ORPE offset, 2-434
 ?ORPL value, 2-434
 ?ORPN offset, 2-434
 ?ORPP offset, 2-434
 ?ORPS offset, 2-434
 ?ORPT offset, 2-434
 ?ORSC offset, 2-432-2-433
 ?ORVR value, 2-66-2-67
 OS abbreviation, 2-2
 ?OSID offset, 2-432
 ?OSIL offset, 2-432
 ?OSLN value, 2-432
 ?OSN2 and ?OSN3 offsets, 2-432-2-433
 ?OSNF offset, 2-432
 ?OSNG value, 2-431
 ?OSNL value, 2-431
 ?OSNN offset, 2-432
 ?OSNO value, 2-431
 ?OSNP offset, 2-432
 ?OSNQ offset, 2-432
 ?OSNR offset, 2-432
 ?OSNT offset, 2-432
 ?OSOL offset, 2-432
 ?OSPI offset, 2-425-2-426
 other file types, getting status of, 2-168
 output, restarting, 2-40
 overhead, pipe, 2-416
 overlay
 exiting from an, 2-510
 loading an, 2-511
 releasing an, 2-509, 2-513
 overrun, timing, 2-790
 ?OVEX system call, 2-509
 ?OVKIL system call, 2-510
 ?OVL0D system call, 2-511-2-514
 ?OVREL system call, 2-513
 owner of a port, finding the, 2-308

P

?PACDEV value, 2-555, 2-556
 packet address and parameters, 1-3
 pages
 shared, flushing to disk, 2-103, 2-162, 2-643
 undedicated memory, 2-204
 unwiring, 2-779
 wiring, 2-76
 palette, 2-223
 ?PALW value, 2-417
 parity setting, field mask, 2-197
 partition, shared, 2-718
 partition size, changing a process's, 2-787

PARU file, A-39, A-40
 PARU.16.SR file, 1-3
 PARU.32.SR file, 1-3, 1-8, 1-9
 PARU_LONG.SR file, 1-3, 1-8
 Pascal language, 1-10
 passing
 connection, 2-516, 2-523
 control from one program to another, 2-33
 PASSTHRU mode, 2-115
 password
 data encryption, 2-590
 encrypting a, 2-554
 length, 2-590
 path, task execution, 2-278
 pathname
 complete, 2-205
 complete, of generic file, 2-239
 getting a file's complete, 2-32
 process or program, 2-222
 remote host, 2-639
 window, 2-226, 2-569
 ?PBATCHP value, 2-555, 2-556
 ?PBCHPRV value, 2-555, 2-556
 ?PBLKS offset, 2-561, 2-904
 ?PBLMEM value, 2-555, 2-556
 ?PBLT value, 2-597, 2-705
 ?PBRK offset, 2-30
 ?PBRK value, 2-538
 ?PBWSS value, 2-555, 2-556
 PBX support, callout, 2-198
 ?PCAD offset, 2-526-2-527, 2-597, 2-598,
 2-705
 ?PCAL offset, 2-537, 2-540, 2-543
 ?PCHPRI value, 2-555, 2-556
 ?PCHTYP value, 2-555, 2-556
 ?PCHUSER value, 2-555, 2-556
 ?PCHWSSL value, 2-556
 ?PCL... offsets and values, 2-515
 ?PCLASS system call, 2-514
 ?PCNSPRV value, 2-555, 2-556
 ?PCNX system call, 2-516
 ?PCOMMNT value, 2-555
 ?PCON offset, 2-537, 2-540
 ?PCS1-?PCS8 offsets, 2-526-2-527
 ?PDBLOCY value, 2-557
 ?PDEL value, 2-409, 2-412
 ?PDESLN value, 2-561, 2-565, 2-904, 2-907,
 2-909
 ?PDFP offset, 2-537, 2-541
 ?PDIR offset, 2-537, 2-539
 ?PDISKLM value, 2-554
 ?PDLOCY value, 2-557
 ?PDMP value, 2-539
 PED, 2-217
 .PER directory, 2-60
 peripheral directory, 2-60
 permanent file attribute, 2-653
 permitting, access to a protected file, 2-519
 ?PFADW offset, 2-520, 2-702
 ?PFAL offset, 2-557
 ?PFBI value, 2-554
 ?PFBS value, 2-538
 ?PFBY value, 2-557
 ?PFCRE value, 2-553
 ?PFDA value, 2-538
 ?PFDB value, 2-84, 2-538
 ?PFDEL value, 2-553
 ?PFDL offset, 2-554, 2-557
 ?PFDLL value, 2-548
 ?PFDP offset, 2-554, 2-557
 ?PFER offset, 2-557
 ?PFEX value, 2-538
 ?PFFC offset, 2-552, 2-553
 ?PFFD offset, 2-554
 ?PFFLG offset, 2-520, 2-701-2-702
 ?PFFO value, 2-520, 2-701-2-702
 ?PFIAC value, 2-552, **2-553**
 ?PFIH offset, 2-520, 2-702
 ?PFLB value, 2-552
 ?PFLE value, 2-553
 ?PFLG offset, 2-84, 2-537-2-540, 2-544
 ?PFLNG value, 2-520, 2-702
 ?PFNF offset, 2-552, 2-553
 ?PFPID offset, 2-520, 2-702
 ?PFPM value, 2-538, 2-540
 ?PFPP value, 2-538

?PFPR offset, 2-552, 2-553
 ?PFR1 offset, 2-552, 2-553
 ?PFR3 offset, 2-557
 ?PFRDF value, 2-553
 ?PFREN value, 2-553
 ?PFRNG offset, 2-520, 2-702
 ?PFRP value, 2-75, 2-539
 ?PFRS value, 2-75, 2-539
 ?PFRV offset, 2-552, 2-553
 ?PFRW value, 2-520, 2-702
 ?PFSE value, 2-554
 ?PFTAC value, 2-552, **2-553**
 ?PFUFD value, 2-553
 ?PFUN offset, 2-552, 2-553
 ?PFVER value, 2-557
 ?PFW value, 2-554
 ?PFXP value, 2-538, 2-545
 ?PHRDPRV value, 2-556
 physical block I/O, 2-210, 2-405, 2-525, 2-592
 ?PICCFN value, 2-554
 ?PICROG value, 2-554
 PID
 and global port number association, 2-219
 getting information about, 2-517
 of a process's father, getting the, 2-79
 returning active, 2-217
 translating a, 2-769
 virtual, 2-266
 ?PIDS system call, 2-517-2-518
 ?PIECE_PKT... offsets and values,
 2-347-2-348
 ?PIFG offset, 2-416-2-417
 ?PIFP offset, 2-537, 2-541
 ?PILN value, 2-416
 ?PILRP offset, 2-518
 ?PILTH value, 2-518
 ?PIMXP offset, 2-518
 ?PINTDIR value, 2-556
 PIO instructions, 2-151, 2-271
 ?PIPC offset, 2-537, 2-539, 2-543
 ?PIPD offset, 2-416-2-417
 pipe extension packet, 2-406, **2-416**
 pipe file, creating a, 2-68
 pipe length, maximum, 2-410
 pipe size, offset ?IMRS, 2-416
 pipes, overhead, 2-416
 ?PIPI offset, 2-416-2-417
 ?PIPR offset, 2-518
 ?PIRS offset, 2-416-2-417
 ?PIRV offset, 2-416-2-417
 PIT device, 2-151, 2-271
 ?PITI offset, 2-416-2-417
 ?PITOT offset, 2-518
 pixel map
 changing colors, 2-234
 deleting, 2-228
 ID, 2-226
 manipulating, 2-223
 related palette, 2-234
 ?PKR0 value, **2-106-2-112**, 2-553
 ?PKR1 value, 2-121, 2-122, 2-694
 PL/I language, 1-10
 ?PLFP offset, 2-537, 2-541
 ?PLOGON value, 2-554
 ?PLTH value, 2-537, 2-546-2-548
 ?PMCTS value, 2-556
 ?PMDIS offset, 2-562, 2-565
 ?PMDSEN offset, 2-562, 2-565
 ?PMEM offset, 2-537, 2-539, 2-543
 ?PMGSYS value, 2-555, 2-556
 ?PMODPRV value, 2-555, 2-556
 ?PMTPF system call, 2-160, **2-519-2-520**
 ?PMXSONS value, 2-554, 2-556
 ?PMYSONS value, 2-555, 2-556
 ?PN1FLG value, 2-557
 ?PN2FLG value, 2-557
 ?PNAME system call, 2-521-2-524b
 ?PNBLMEM value, 2-555, 2-556
 ?PNBWSS value, 2-555, 2-556
 ?PNCRYPT value, 2-554
 ?PNM offset, 2-537, 2-539
 ?PNVR value, 2-417
 ?POBLOCY value, 2-557
 ?POFP offset, 2-537, 2-541
 point-to-point station, 2-715
 pointer
 byte, 1-5

- device, controlling input from a, 2-567
- event, 2-191
- file, 2-220, 2-610, 2-707
- frame, 1-2
- poll, address and list, 2-664-2-667
- ?POLOCY value, 2-557
- port number
 - global, 2-219, 2-288, 2-289, 2-308, 2-770
 - local, 2-219, 2-770
 - terminal, 2-185
- position, bit, 1-4
- powerfail/restart routine, user device, 2-277
- ?PPASSWD value, 2-554
- ?PPBLT value, 2-526
- ?PPCR offset, 2-537, 2-540
- ?PPDPMGR value, 2-555, 2-556
- ?PPRCINF value, 2-556
- ?PPRI offset, 2-537, 2-539, 2-543
- ?PPRNBLK value, 2-555, 2-556
- ?PPRV offset, 2-537, 2-540
- ?PPSUPP value, 2-555, 2-556
- ?PPWDPRV value, 2-555, 2-556
- ?PQBLOCY value, 2-557
- ?PQLOCY value, 2-557
- ?PRBB offset, 2-526-2-529
- ?PRCL offset, 2-260, 2-526-2-527, 2-597, 2-598, 2-705
- ?PRCNX system call, 2-523-2-524b
- ?PRCRYPT value, 2-557
- ?PRDB system call, 2-525-2-526
- PREDITOR utility program, 2-552, 2-744
- ?PRES offset, 2-597, 2-598, 2-705
- ?PRHRDPR value, 2-557
- ?PRI system call, 2-530
- prime meridian, 2-247, 2-400
- priority
 - changing a process, 2-531
 - changing a task, 2-530
 - getting calling task, 2-399
- ?PRIPR system call, 2-531-2-534b
- privilege state, 2-744
- privileges, access control, 2-245
- ?PRKIL system call, 2-533
- ?PRNH offset, 2-16, 2-260, 2-526-2-527, 2-596, 2-597, 2-598, 2-705
- ?PRNL offset, 2-597, 2-598
- ?PROC extension packet, 2-545-2-548
- ?PROC functions
 - creating offspring, 2-544
 - sending a CLI-like command line, 2-543
 - setting maximum CPU time, 2-544
 - setting the working set size, 2-545
- ?PROC system call, 2-240, 2-534-2-534h
 - examples of, A-9, A-48
- procedure, chaining to a new, 2-595
- process
 - address space, remapping a, 2-353
 - blocking a, 2-26
 - changing priority of a, 2-531
 - class and locality, 2-514
 - communication, operator/current, 2-437
 - creating a, 2-534
 - getting the PID of a father, 2-79
 - getting the virtual PID of a, 2-266
 - location, 2-28
 - management system calls (names of), 2-4-2-5
 - name
 - full, 2-265, 2-521
 - locating a, 2-28
 - partition size, changing, 2-787
 - pathname, getting a, 2-222
 - priority values, 2-532
 - returning status information, 2-560
 - runtime statistics, 2-648
 - son, 2-696
 - status information, extended, 2-900
 - synchronizing, 2-305
 - terminal, 2-771
 - terminating a, 2-29, 2-754
 - termination
 - code, 2-299
 - message, 2-295-2-297, 2-297, 2-635
 - termination message, 16-bit B-type or C-type, 2-304.6
 - termination messages
 - B-type, 2-304-2-304.8
 - C-type, 2-304-2-304.8
 - type, changing a, 2-75
 - unblocking a, 2-776
 - username, 2-265
 - waiting for another, 2-845
- PROCESS command, 2-255
- Process Environment Display utility program, 2-217
- process types, 2-297, 2-304

processor
 class assignments, logical, 2-362
 identification, unique hardware, 2-263

profile requests and functions, 2-551-2-552

?PROFILE system call, 2-547, 2-548,
 2-551-2-552

program
 assembly language example, 1-5-1-9
 chaining to, 2-33
 construction and execution, 1-8
 file, loading a, 2-637
 getting a pathname, 2-222
 returning (.PR) filename, 2-640
 sample sets, iv, 1-5-1-9, x

protected file, 2-519

protected shared file, 2-701
 opening a, 2-701

protecting, a task from being redirected, 2-757

protocol data-link control characters, BSC,
 2-674

?PRPSSWD value, 2-557

?PRRAPRV value, 2-517, 2-555, 2-556

?PRRDY system call, 2-558

?PRSFTPR value, 2-557

?PRSUS system call, 2-559

?PSAL value, 2-562, 2-563

?PSCH offset, 2-562, 2-564

?PSCPL offset, 2-562, 2-564

?PSCW offset, 2-562, 2-564

?PSEN offset, 2-562, 2-563

?PSEX offset, 2-562, 2-564, 2-565

?PSF2-?PSF5 offsets, 2-562-2-564

?PSFA offset, 2-562, 2-565

?PSFL offset, 2-562, 2-563

?PSFP offset, 2-562, 2-563

?PSFTPRV value, 2-556

?PSHRP value, 2-564, 2-906

?PSHSH offset, 2-561, 2-904

?PSHST offset, 2-561, 2-904

?PSHSZ offset, 2-561, 2-904

?PSIH offset, 2-562, 2-565

?PSLFA offset, 2-562, 2-565

?PSLTH value, 2-562

?PSMX offset, 2-562, 2-565

?PSNM offset, 2-537, 2-539

?PSNR offset, 2-562, 2-563

?PSNS offset, 2-562, 2-563

?PSOPIO value, 2-554, 2-556

?PSPD offset, 2-562, 2-564

?PSPH offset, 2-562, 2-564

?PSPP value, 2-563, 2-905

?PSPR offset, 2-562, 2-564

?PSPRST offset, 2-561, 2-904

?PSPV offset, 2-562, 2-564

?PSQF offset, 2-562, 2-563

?PSRH offset, 2-562, 2-564

?PSSL offset, 2-562, 2-565

?PSSN offset, 2-562, 2-563

?PSSP value, 2-563, 2-905

?PSST offset, 2-562, 2-563

?PSTAT system call, 2-560-2-566

?PSTI offset, 2-16, 2-526-2-529, 2-597, 2-598,
 2-704, 2-705

?PSTO offset, 2-526-2-527, 2-597, 2-598,
 2-705

?PSUSER value, 2-555, 2-556

?PSWM offset, 2-562, 2-565

?PSWS offset, 2-562, 2-565

?PSXPT value, 2-564, 2-906

PTE abbreviation, 2-781

?PTRDEV_EVENTS... values, 2-578, 2-586

?PTRDEV_GEN_EVENT... offsets and values,
 2-585-2-586

?PTRDEV_GENERATE_EVENT function,
 2-568, **2-584-2-586**

?PTRDEV_GENERATE_EVENT_PKTID value,
 2-586

?PTRDEV_GET_LOC... offsets, 2-587-2-588

?PTRDEV_GET_PTR_LOCATION function,
 2-568, 2-571, 2-587

?PTRDEV_GET_PTR_STATUS function, 2-568,
 2-581

?PTRDEV_GET_TABLET_LOCATION
 function, 2-568, 2-571, 2-588-2-589

?PTRDEV_GET_TABLOC... offsets and values,
 2-588

?PTRDEV_GSTATUS... offsets and values,
 2-581-2-584

?PTRDEV_LAST_EVENT function, 2-568,
 2-571, **2-577**

?PTRDEV_LEVENT... offsets and values, 2-577-2-578

?PTRDEV_PKT... offsets and values, 2-569-2-571

?PTRDEV_PTR_SHAPE... values, 2-580-2-581, 2-582

?PTRDEV_PTR_STATE... values, 2-580-2-581

?PTRDEV_SET_DELTA function, 2-568, 2-573, **2-576-2-577**

?PTRDEV_SET_DELTA... offsets and values, 2-576-2-577

?PTRDEV_SET_DELTA_LEN value, 2-576

?PTRDEV_SET_EVENTS function, 2-568, 2-573

?PTRDEV_SET_EVTS... offsets and values, 2-573, 2-574-2-576

?PTRDEV_SET_POINTER function, 2-568, **2-579-2-580**

?PTRDEV_SET_PTR... offsets, 2-579-2-580

?PTRDEVICE functions

- controlling the operation of the pointer, 2-579-2-580
- dead tablet space, 2-574
- generating a pointer event, 2-584-2-586
- getting
 - information about the last pointer event, 2-577
 - pointer status, 2-581
 - status of the pointer device, 2-587
- getting the tablet status, 2-588
- moving the pointer, 2-584-2-586
- selecting pointer events, 2-573-2-575
- specifying a pointer delta, 2-576

?PTRDEVICE system call, 2-191, **2-567-2-568**

- example of, A-61

?PTWO value, 2-417

?PUDAH offset, 2-603

?PUDAL offset, 2-603

?PUDCN offset, 2-603

?PUDFW offset, 2-603

?PUDLT value, 2-603

?PUIPCS value, 2-555, 2-556

?PUL_MAX_NAMES value, 2-388, 2-892

?PUL_PKT... offsets and values, 2-388, 2-892

?PUNM offset, 2-537, 2-540

?PUSPR offset, 2-561, 2-904

PVC circuit connections, see ?CONINFO, 2-58.9

?PVCNPRV value, 2-555, 2-556

?PVDV value, 2-542

?PVEX value, 2-542

?PVIP value, 2-542

?PVPC value, 2-542

?PVPP value, 2-542

?PVPR value, 2-542

?PVSP value, 2-542

?PVSU value, 2-542

?PVTY value, 2-542

?PVUI value, 2-542

?PVWM value, 2-542

?PVWS value, 2-542, 2-545

?PWBP offset, 2-591

?PWDCRYP system call, 2-554, **2-590-2-591**

?PFW offset, 2-591

?PWLO offset, 2-591

?PWMI offset, 2-537, 2-541, 2-545

?PWOW value, 2-591

?PWRB system call, 2-213, **2-525-2-526**, 2-592

?PWRV offset, 2-591

?PWSO value, 2-556

?PWSS offset, 2-537, 2-540, 2-545

?PWSZ value, 2-591

?PXCPU offset, 2-546-2-548

?PXFLG offset, 2-546-2-548

?PXLE value, 2-545, 2-546-2-548

?PXLLOC offset, 2-546-2-548

?PXPAG offset, 2-546-2-548

?PXPGI offset, 2-546-2-548

?PXPGN offset, 2-546-2-548

?PXPUN offset, 2-546-2-548

?PXRES offset, 2-546-2-548

?PXRS0 and ?PXRS1 offsets, 2-546-2-548

?PXSID value, 2-548

?PXSPI offset, 2-546-2-548

?PXULOC offset, 2-546-2-548

?PXUPID offset, 2-546-2-547

Q

QSYM.F77.IN file, A-40

queue
 name, locating a, 2-28
 removing tasks from a, 2-90
 task manager, 2-294

queues
 batch, document names, 2-122
 print, document names, 2-122

R

radix, v, xi

?RCALL system call, -594
?RCHAIN system call, 2-595
?RCID value, 2-433
?RDAC value, 2-781
?RDB system call, 2-596-2-601
?RDUDA system call, 2-602-2-603

re-enabling
 control-character terminal interrupt, 2-334
 relative terminal, 2-667, 2-685

re-establishing, connection, 2-516, 2-523

?READ system call, 2-191, **2-604**
 examples of, A-12, A-19, A-22-A-23, A-33, A-67

read/write access to memory, 2-782

reading
 allocated blocks, 2-23
 block I/O, 2-596
 device characteristics, 2-177
 error message file, 2-99
 record I/O, 2-604
 shared-page, 2-704
 task message from the process terminal, 2-771
 time-of-day conversion data, 2-645
 user data area, 2-602-2-604

reading
 task, 2-281
 task status word, 2-282
 tasks of a specified priority, 2-558

real-time clock, 2-313

?REC system call, 2-627
 example of, A-34

receive/continue call, 2-714

receiving
 after sending an IPC message, 2-309
 information over a BSC line, 2-709
 interrupt service message, 2-290

intertask message, 2-627
 intertask message without waiting, 2-628
 IPC message, 2-295

?RECNW system call, 2-628

record I/O, 2-604
 performing, 2-604
 writing, 2-604, 2-844

?RECREATE system call, 2-629

recreating, a file, 2-629

rectangle, clip, 2-223

redirecting, task, 2-774
 execution path, 2-278
 protection, 2-757

register
 floating-point status, 2-285
 stack, 2-314

Related manuals, ix

relative terminal, 2-667, 2-685
 disabling a, 2-667
 re-enabling a, 2-667, 2-685

?RELEASE system call, 2-630

releasing
 initialized logical disk, 2-630
 job processor, 2-322
 overlay, 2-509, 2-513
 resource, -594
 shared page, 2-643

?REM value, 2-168

remapping, a process's address space, 2-353

remote host, 2-639
 process and queue name on, 2-28

removing
 permanent file attribute, 2-653
 tasks from a queue, 2-90
 user device, 2-304.9

?RENAME system call, 2-632-2-632.2

renaming, a file, 2-632

reporting, index, 1-9

request, profile, 2-551

?RESCHED system call, 2-633

rescheduling
 disabling task, 2-88
 task, 2-280
 time slice, 2-633

reserved symbols, 1-5

reset MRC routes, current, 2-58

?RESIGN system call, 2-634

- resource
 - acquiring a, -594
 - acquiring a new, 2-328
 - base of the current, 2-186
 - calling, 2-328
 - releasing a, -594
- resources, system calls, 2-12
- restoring, the previous environment, 2-778
- return
 - error and normal, 1-2
 - normal, 1-2
- ?RETURN system call, 1-2, **2-635**
 - examples of, 1-6-1-7, A-4, A-6, A-8-A-9, A-12, A-15, A-19, A-23, A-27, A-29, A-33-A-34, A-37-A-38, A-43, A-48, A-59
- returning
 - active PIDs, 2-217
 - class scheduling statistics, 2-45
 - code and text (error), 2-683
 - complete pathname of generic file, 2-239
 - error code and text, 2-683
 - extended status information on a process, 2-900
 - from a process, 2-635
 - global port number, 2-288
 - LEF mode status, 2-352
 - logical disk information, 2-340
 - number of undedicated memory pages, 2-204
 - OS-format internal time, 2-313
 - PID associated with a global port number, 2-219
 - process statistics, 2-560
 - program (.PR) filename, 2-640
 - stack frame information, 2-807
 - status information on a process, 2-560
 - status of a task, 2-756, 2-777
 - text and code (error), 2-683
 - unique task identifier, 2-688, 2-690, 2-777
- ?RFAB value, 2-298, 2-635
- ?RFCF value, 2-298, 2-635
 - example of, 1-7
- ?RFEC value, 2-298, 2-635
 - example of, 1-7
- ?RFER value, 2-298, 2-635
 - example of, 1-7
- ?RFWA value, 2-298, 2-635
- ring
 - base address, 2-565
 - field, 2-289
 - loading, stopping, 2-642
 - lower, 2-353
- ?RINGLD system call, 2-160, **2-637-2-641**
 - example of, A-14
- RINGLOAD.SR sample program, A-2, **A-14-A-18**
- ?RLFRC offset, 2-630
- RMA access, 2-517
- ?RNAME system call, 2-639
- ?RNGBP offset, 2-641
- ?RNGLB offset, 2-641
- ?RNGNM offset, 2-641
- ?RNGPL value, 2-641
- ?RNGPR system call, 2-640-2-641
- ?RNGST system call, 2-642
- routine, power, fail/restart, 2-272
- ?RPAGE system call, 2-643-2-644
- ?RSID value, 2-435
- RTC device, 2-151, 2-271
- ?RTDS value, 2-409, 2-412, 2-608, 2-609
- ?RTDY value, 2-409, 2-608, 2-609
- ?RTFX value, 2-409, 2-608, 2-609
- ?RTODC system call, 2-645-2-647
- ?RTODC_PKT... offsets and values, 2-646-2-647
- ?RTUN value, 2-409, 2-608, 2-609
- ?RTVB value, 2-409, 2-608, 2-609
- ?RTVR value, 2-409, 2-608, 2-609
- runtime process statistics, 2-648
 - getting, 2-648
- RUNTIME.SR sample program, A-1, **A-11-A-13**
- ?RUNTM system call, 2-648
 - example of, A-11
- ?RVBPL value, 2-783
- ?RVBPX value, 2-783
- RVI character, 2-676
- ?RVWPL value, 2-783

S

- S operator, 1–4
- ?SACK value, 2–711
- ?SACL system call, 2–213, **2–650–2–652**
- ?SACP offset, 2–165–2–170
- ?SAFM offset, 2–527
- ?SAFM value, 2–598
- ?SAK0 and ?SAK1 values, 2–711
- sample programs, 1–5–1–9
- ?SASC value, 2–671, 2–677
- ?SATR system call, 2–653–2–654
- ?SAVS value, 2–693
- ?SBER offset, 2–686, 2–687
- ?SBIAS system call, 2–655
- ?SBSC value, 2–672
- ?SBUL offset, 2–721
- ?SBUP offset, 2–710, 2–712, 2–721, 2–723
- ?SBYC offset, 2–710, 2–712, 2–721, 2–723
- ?SBYM offset, 2–710, 2–713, 2–721, 2–723
- scalar date value, converting a, 2–31
- scalar time value, converting a, 2–74
- scaled space on a tablet, 2–572
- scheduler, system, 2–27
- scheduling
 - disabling task, 2–93
 - enabling multitask, 2–102
- ?SCHN offset, 2–670, 2–671
- ?SCHR system call, 2–656–2–657
- ?SCIT value, 2–671
- ?SCLOSE system call, 2–658–2–659
- ?SCON value, 2–722
- SCP boot clock, 2–401
- SCP device, 2–151, 2–271
- ?SCPS offset, 2–168
- ?SCRC value, 2–671
- screen management extension, 2–612
- ?SCSH offset, 2–167, 2–168
- ?SCSL offset, 2–167, 2–168
- ?SDAC value, 2–712, 2–716, 2–722
- ?SDAD offset, 2–710, 2–713
- ?SDAY system call, 2–660
- ?SDBL system call, 2–661
- ?SDCN value, 2–188, 2–662
- ?SDCU offset, 2–165
- ?SDDN value, 2–188, 2–662
- ?SDEH offset, 2–168
- ?SDEL offset, 2–168
- ?SDET value, 2–723
- ?SDIS value, 2–723
- ?SDLM system call, 2–662–2–663
- ?SDMD value, 2–671
- ?SDPOL system call, 2–664–2–666
- ?SDPP value, 2–671
- ?SDPR value, 2–671
- ?SDRT system call, 2–667–2–668
- ?SDSC value, 2–671
- ?SDTI value, 2–188, 2–662
- ?SDTO value, 2–188, 2–662
- ?SDTP value, 2–188, 2–662
- search list
 - getting contents of a, 2–203
 - setting the, 2–695
- ?SEBC value, 2–671
- ?SEBL system call, 2–669–2–678
- ?SECHR system call, 2–41, **2–679–2–680**
- sector I/O, modified, 2–20, 2–25, 2–216
- ?SEFH offset, 2–167, 2–168
- ?SEFL offset, 2–167, 2–168
- ?SEFM offset, 2–167, 2–168
- ?SEFW offset, 2–167, 2–168
- ?SEID value, 2–431
- select address, 2–665
- select–address pair, 2–664
- ?SELN value, 2–670
- ?SEND system call, 2–681–2–682
 - example of, A–27
- sending
 - information over a BSC line, 2–720
 - IPC message, 2–305, 2–309
 - terminal message, 2–681
- ?SEOT value, 2–723
- ?SEPR value, 2–671, 2–677
- sequences, keyboard interrupt, 2–332

?SERMSG system call, 2-683-2-684

?SERT system call, **2-667**, 2-685

?SERVE system call, 2-685

server

- becoming a, 2-685
- becoming a customer of, 2-56
- resigning as a, 2-634

server/customer relationship, 2-34, 2-57

- disconnecting a, 2-80, 2-92
- terminating a, 2-72

service

- message, interrupt, 2-290
- routine
 - device interrupt handler, 2-275
 - fast device interrupt handler, 2-156

session, connection types, 2-58.9

set/get class ID code, 2-36-2-56

setting

- access control list, 2-650
- bias factor values, 2-655
- binary I/O on a pipe, 2-611
- bit, 1-4
- class IDs, 2-36
- class matrix, 2-51
- data channel map, 2-729
- default access control list, 2-77
- delimiter table, 2-662
- device time-out value, 2-733
- execute-protection status, 2-141-2-142
- extended device characteristics, 2-679
- file-pointer position, 2-707
- IPC no wait, **2-409**, 2-611
- logical processor class assignments, 2-362
- maximum size for a control point directory, 2-58.26
- permanent file attribute, 2-653
- search list, 2-695
- system
 - calendar, 2-660
 - clock, 2-732
 - identifier, 2-719
 - time of day, 2-400, 2-732

?SFAH offset, 2-167, 2-168

?SFAL offset, 2-167, 2-168

?SGES system call, 2-686-2-687

?SGLN value, 2-686

shared

- access, 2-699
- access file
 - closing a, 2-658
 - opening a, 2-405
- file, 2-659, 2-699
 - protected, 2-701
- file memory pages, flushing to disk, 2-103
- page, 2-643
 - flushing a disk, 2-162
 - read, 2-704
 - partition, 2-243, 2-718

?SHC0 offset, 2-198

?SHC0 value, 2-335

?SHCO value, 2-180

?SHFS offset, 2-167

?SHOP value, 2-409, **2-414**

/SHR switch, 2-180

?SHSP value, 2-672

?SIDX offset, 2-167, 2-168

?SIEX value, 2-693

signaling

- another task, 2-688, 2-690
- mechanism, interprocess, 2-688, 2-690

signaling mechanism, interprocess, 2-691, 2-845

significant bits, least and most, 1-5

?SIGNL system call, 2-688-2-689, 2-691, 2-845, 2-848

?SIGWT system call, 2-690-2-691

?SIID offset, 2-693

?SILN offset, 2-693

?SIMM offset, 2-693

?SINFO system call, 2-692-2-692b

?SINT value, 2-723

?SIOS offset, 2-693

?SIPL value, 2-693

?SIRL offset, 2-721, 2-723

?SIRN offset, 2-693

?SIRS offset, 2-693

?SITB value, 2-711, 2-716, 2-722

size, shared partition, 2-243

?SLAU offset, 2-167, 2-168

?SLBC value, 2-739-2-740

?SLCON value, 2-739

?SLCSU value, 2-739-2-740

?SLDS value, 2-739

?SLEC value, 2-739-2-740

?SLES value, 2-739

?SLEX value, 2-739
 ?SLFL value, 2-739
 ?SLIST system call, 2-695
 @SLNx device, 2-669
 ?SLON value, 2-739
 ?SLRC value, 2-671, 2-677
 ?SLRE value, 2-739
 ?SLRF value, 2-739-2-740
 ?SLRS value, 2-739
 ?SLSEX value, 2-739
 ?SLSF value, 2-739-2-740
 ?SLSP value, 2-739
 ?SLST value, 2-739-2-740
 ?SLSU value, 2-739
 ?SLTE value, 2-739
 ?SLTH value, 2-165-2-170
 ?SMCH offset, 2-537, 2-541, 2-544
 ?SMDI offset, 2-670, 2-672
 ?SMIL offset, 2-167, 2-168
 ?SMSH offset, 2-167
 ?SMSL offset, 2-167
 ?SNAK value, 2-711
 ?SNID value, 2-673, 2-723
 ?SNKC offset, 2-686, 2-687
 ?SNPR value, 2-671, 2-677
 ?SOAL offset, 2-697-2-698
 ?SOFP offset, 2-697-2-698
 ?SOFW offset, 2-697-2-698
 SOH character, 2-676
 ?SOHB value, 2-711, 2-716, 2-722
 ?SOKEY offset, 2-697-2-698
 ?SOLTH value, 2-697
 son process, 2-696
 SON.SR sample program, A-1, **A-9-A-10**
 ?SONEN offset, 2-697-2-698
 ?SONS system call, 2-696-2-698
 ?SOPEN system call, 2-699-2-700
 ?SOPN offset, 2-165, 2-166, 2-167, 2-168
 ?SOPPF system call, 2-160, **2-701-2-702**
 ?SOPR value, 2-671, 2-677
 ?SORP offset, 2-697-2-698
 ?SOSON offset, 2-697-2-698
 ?SOSP offset, 2-697-2-698
 ?SPAGE system call, 2-704
 ?SPAR value, 2-618
 SPEAK.SR sample program, A-1, A-7
 special key characteristics, 2-191
 ?SPET value, 2-712
 ?SPLR value, 2-711, 2-716
 ?SPNH offset, 2-166
 ?SPNK value, 2-712
 ?SPNL offset, 2-166
 ?SPOS system call, 2-707-2-708
 example of, A-22
 ?SPRO value, 2-705
 ?SPRV value, 2-712
 ?SPTM offset, 2-73
 ?SR32 value, 2-693
 ?SRCV output values, 2-716
 ?SRCV system call, 2-673, **2-709-2-710**
 ?SRES offset, 2-721, 2-723
 ?SRID value, 2-673, 2-711, 2-713
 ?SRVI value, 2-711
 ?SSHPT system call, 2-718
 ?SSID system call, 2-719
 ?SSIL offset, 2-710, 2-712
 ?SSIN offset, 2-693
 ?SSIS offset, 2-710-2-712, 2-721-2-723
 ?SSLR value, 2-711, 2-716
 ?SSND system call, 2-673, **2-720**
 ?SSNL value, 2-710, 2-721
 ?SSTI offset, 2-669-2-672
 ?SSTO offset, 2-687
 ?SSTS offset, 2-165-2-170
 stack
 frame information, 2-807
 register, 2-314
 unwinding the, 2-778
 ?STAH offset, 2-165-2-170, 2-214
 ?STAL offset, 2-165-2-170, 2-214
 standard format for system calls, 1-2
 starting a histogram, 2-286, 2-393, 2-810
 state save area, 2-284

- station
 - identification, **2-672-2-673**, 2-713, 2-725
 - multipoint and point-to-point control, 2-715
 - multipoint tributary, 2-716
- statistics
 - BSC error, 2-686
 - returning class scheduling, 2-45
 - runtime process, 2-648
- status
 - information about a process, extended, 2-900
 - information on a process, returning, 2-560
 - register, floating-point, 2-285
 - word
 - controller, 2-528-2-529
 - task, 2-282
- ?STCH offset, 2-165-2-168
- ?STCL offset, 2-165-2-170
- ?STIM offset, 2-165-2-170
- ?STMAP system call, 2-729-2-731
- ?STMH offset, 2-165-2-170
- ?STML offset, 2-165-2-170
- ?STOC offset, 2-670, 2-672, 2-677
- ?STOD system call, 2-732
- ?STOM system call, 2-733-2-734
- stop bits, stop bit mask, 2-197
- ?STOV offset, 2-710, 2-713, 2-721, 2-723
- streaming mode of tape I/O, 2-215, 2-412
- ?STTD value, 2-723
- ?STTO offset, 2-686
- STX character, 2-676
- ?STXB value, 2-711, 2-716, 2-722
- ?STYP offset, 2-165-2-167, 2-168
- Superprocess
 - mode, 2-735
 - privilege, 2-745
- Superuser
 - mode, 2-737
 - privilege, 2-745
- ?SUPROC system call, 2-735-2-740
- ?SUS system call, 2-736
- ?SUSER system call, 2-737-2-740
- suspending
 - task, 2-27, 2-85, 2-283, 2-736, 2-809
 - tasks of a specified priority, 2-559
- ?SWAK value, 2-712, 2-723
- ?SYFBM offset, 2-743
- ?SYFLT offset, 2-743
- ?SYLEN value, 2-740, 2-743
- ?SYLID value, 2-743
- ?SYLOG, viewing output, 2-741
- ?SYLOG system call, 2-360, **2-739-2-740**
- symbol table file, 2-256, 2-261
- symbolic debugger utility program, 1-3
- symbols, reserved, 1-5
- SYN character, 2-677
- synchronized logical disk unit image, 2-887
- synchronizing
 - LDU images, 2-385
 - processes, 2-305
- SYSID file, A-39, A-40
- SYSID.32.SR file, 1-8
- :SYSLOG file, **2-360**, 2-739-2-740
- SYSLOG record formats, iii, ix, B-1
- ?SYSPRV system call, 2-744
- ?SYSPRV_... offsets and values, 2-745-2-746
- system
 - area, 2-213
 - calendar, 2-660
 - call
 - implicit, A-39
 - log file, 2-357
 - clock, 2-187, 2-258, 2-645
 - frequency of the, 2-201
 - setting the, 2-732
 - identifier
 - getting the, 2-244
 - setting the, 2-719
 - log file
 - entering an event in the, 2-360
 - event codes, B-1
 - manipulating the, 2-739
 - record formats, B-1
 - scheduler, 2-27
- system calls, 1-1
 - 16-bit process (names of), 2-14
 - AOS/RT32 (names of), 2-3
 - AOS/VS system resources (names of), 2-12
 - class scheduling (names of), 2-12
 - connection management (names of), 2-11, 2-13
 - debugging (names of), 2-7
 - file creation and management (names of), 2-6
 - file input/output (names of), 2-7-2-8
 - interprocess communications (names of), 2-10
 - logging, 2-357
- system calls, 1-1 (continued)
 - memory management (names of), 2-3

- multiprocessor management (names of), 2-11, 2-13
- multitasking (names of), -9-10
- process management (names of), 2-4-2-5
- standard format for, 1-2
- windowing (names of), 1-9

system log, record formats, iii, ix

system log, writing records, B-1

system log file, iii, ix

system log file, B-1

System Manager privilege, 2-745

system resources, system calls, 2-12

SYSTEM_CALL_SAMPLES subdirectory of :UTIL, A-1

?SYSULEN value, 2-743

T

?T32T value, 2-300, 2-301

table

- delimiter, 2-188, 2-416, 2-662
- file, symbol, 2-261
- map definition, 2-153, 2-154, 2-272, 2-731

tablet

- dead space on, 2-572
- digitize option for, 2-572, 2-574, 2-579
- digitizing, 2-571-2-572
- location example, 2-573
- scaled space on, 2-572

?TABR value, 2-300

?TALOCK value, 2-757, 2-774

?TAOS value, 2-300

tape

- block I/O, 2-599
- density values, 2-337
- I/O
 - buffered and streaming modes, 2-215
 - buffered mode of, 2-214
- labeled magnetic, 2-35, 2-38, 2-418
 - forcing end-of-volume, 2-148
- magnetic, 2-212, 2-417

task

- changing priority of a, 2-530
- changing the priority of a, 2-280
- control block, 2-282
- delaying a, 2-85, 2-809
- execution path, 2-278
- extended definition, 2-751
- identifier, 2-399
 - unique, 2-688, 2-690, 2-777

- initiating a, 2-747
- initiation queue, 2-294
- interrupt, 2-278
- killing a, 2-279, 2-331, 2-510
- killing a group, 2-533
- manager, queued, 2-294
- message, reading from the process terminal, 2-771
- multi-scheduling, 2-102
- readying a, 2-281
- redirecting a, 2-278, 2-774
- redirection protection, 2-757
- rescheduling a, 2-88, 2-280-2-281
- returning status of a, 2-756, 2-777
- scheduling, disabling, 2-93
- signaling another, 2-688, 2-690
- status, returning, 2-756, 2-777
- status word, 2-282
- suspending a, 2-27, 2-85, 2-283, 2-736, 2-809
- suspension, 2-281
- terminal interrupt, 2-293
- waiting for another, 2-845

?TASK system call, 1-2, **2-747-2-748**

- examples of, A-29, A-33

tasks

- readying, 2-558
- removing from a queue, 2-90
- suspending, 2-559

?TATH offset, 2-63

?TBCX value, 2-73, 2-300

?TBLT value, 2-63

TCB, 2-282

?TCCX value, 2-300

?TCIN value, 2-299-2-301

?TCTH offset, 2-63

?TEFH offset, 2-260

?TEFM offset, 2-260

?TEFW offset, 2-260

template filename characters, 2-208

?TERM system call, 2-754-2-755

- example of, A-7

terminal

- ANSI-standard, 2-182
- interrupt
 - control-character, 2-333-2-334
 - disabling, 2-403
 - enabling, 2-404
 - task, 2-293
 - waiting for a, 2-335
 - message, 2-681
 - port number, 2-185

- process, 2-771
- relative, 2-667, 2-685
- terminal services, and console line numbers, 2-58.8
- terminating
 - customer/server relationship, 2-72
 - process, 2-29, 2-635, 2-754
- termination
 - code, process, 2-299
 - message
 - 16-bit process, 2-303
 - 32-bit process, 2-302
 - process, 2-297, 2-635
 - routine, user device, 2-276
- termination message, packet structure, B- or C-type, 2-304.5
- text and code error, returning, 2-683
- ?TEXT value, 2-298-2-304.8
- TID
 - and priority of the calling task, getting, 2-399
 - unique, 2-688, 2-690, 2-777
- ?TIDSTAT system call, 2-756
- time
 - current, 2-313
 - date and time zone
 - getting the, 2-247
 - setting the, 2-400
 - Greenwich Mean and Universal, 2-247, 2-400
 - last accessed value, 2-214
 - of day
 - conversion data, reading, 2-645
 - converting to a scalar value, 2-171
 - getting the, 2-258
 - setting the, 2-732
 - overrun, 2-790
 - returning the OS-format internal, 2-313
 - slice, rescheduling a, 2-633
 - value
 - converting a scalar, 2-74
 - format, 2-46
- time-out
 - connect, 2-677
 - value for a device, 2-733-2-735
- ?TIME_PKT... offsets and values, 2-248-2-249, 2-401-2-402
- TIMEOUT.SR sample program, A-2, **A-37-A-38**
- Timer Facility, Virtual, 2-790
- /TLA FILESTATUS command switch, 2-214

- ?TLOCK system call, 2-757-2-758
- ?TLTH value, 2-260
- ?TM6 value, 2-304.2, 2-762
- ?TMPID offset, 2-304.2, 2-762
- ?TMPRV offset, 2-304.2, 2-762
- ?TMRS offset, 2-304.2, 2-762
- ?TMTH offset, 2-63
- ?TMUPD offset, 2-304.2, 2-762
- ?TMYRING value, 2-757, 2-774
- ?TPID system call, 2-769
- ?TPLN value, 2-301
- ?TPORT system call, 2-770
- ?TR32 value, 2-300
- trailer label, 2-38
- ?TRAN value, 2-711, 2-716, 2-722
- transfer modes, magnetic tape, 2-410
- translating
 - local port number to global equivalent, 2-770
 - PID, 2-769
- translation, field, 2-616
- transmitting
 - intertask message, 2-898, 2-899
 - message from an interrupt service routine, 2-157, 2-315
- ?TRAP value, 2-300
- ?TRCON system call, 2-771-2-772
- tributary station, 2-665
 - multipoint, 2-716
- Trojan pointer, 2-781
- ?TRPE offset, 2-198
- ?TRPE value, 2-191, 2-614, 2-680
- ?TRUNCATE system call, **2-773, A-39**
- truncating, a file, 2-259, 2-773
- ?TSELF value, 2-300
- ?TSSP value, 2-756
- ?TSTAT word, 2-282
- ?TSUP value, 2-300, 2-303
- TTD character, 2-677
- TTI device, 2-151, 2-271
- TTO device, 2-151, 2-271
- ?TTY value, 2-181
- ?TUNLOCK system call, 2-774-2-775

U

- ?UBLPR system call, 2-776

- ?UDBM value, 2-153-2-154, 2-273-2-274
- ?UDDA-?UDDP values, 2-153-2-154, 2-273-2-274
- ?UDDRS offset, 2-152, 2-271, 2-272, 2-277
- ?UDDTR offset, 2-152, 2-271, 2-272, 2-276
- ?UDELTH value, 2-153, 2-272
- ?UDFE value, 2-152
- ?UDFX value, 2-152
- ?UDID offset, 2-153-2-154, 2-273-2-274
- ?UDIRL offset, 2-152
- ?UDLE value, 2-271
- ?UDLN value, 2-152, 2-271-2-272
- ?UDLTH value, 2-153, 2-272
- ?UDNS offset, 2-154, 2-274
- UDPR, 2-277
- ?UDRS offset, 2-152, 2-271, 2-272
- UDTR, 2-276
- ?UDVBX offset, 2-152, 2-272
- ?UDVIS offset, 2-152, 2-272
- ?UDVMS offset, 2-152, 2-272
- ?UDVXM offset, 2-152, 2-272
- ?UIDSTAT system call, 2-777
- ?ULI1-?ULI3 values, 2-392, 2-897
- ?ULLN value, 2-392, 2-897
- ?ULN1-?ULN8 offsets, 2-392, 2-897
- ?ULNC offset, 2-392, 2-897
- ?ULPHI offset, 2-392, 2-897
- ?ULPLO offset, 2-392, 2-897
- ?ULR0-?ULR8 offsets, 2-392, 2-897
- unblocking, a process, 2-776
- undedicated memory pages, 2-204
- uniprocessor histogram, 2-393
- unique
 - hardware processor identification, 2-263
 - task identifier, 2-688, 2-690, 2-777
- unit file, getting status of, 2-165
- Universal Time, 2-247, 2-400
- unlocking
 - an object, 2-172
 - whole files, 2-173
- ?UNMR value, 2-374, 2-376

- unshared
 - memory pages, changing, 2-384
 - memory parameters, listing, 2-383
- ?UNWIND system call, 2-778
- unwinding, the stack, 2-778
- ?UNWIRE system call, 2-779
- unwiring, pages, 2-779
- ?UPDATE system call, 2-213, 2-780
- upfront window, 2-574
- UPSC device, 2-151, 2-271
- ?URTB offset, 2-156
- user
 - data area
 - creating a, 2-70
 - reading a, 2-602-2-604
 - writing a, 2-602-2-604, 2-844
 - device, 2-18
 - defining a, 2-269
 - defining a fast, 2-149
 - powerfail/restart routine, 2-277
 - removing a, 2-304.9
 - termination routine, 2-276
 - locality, changing, 2-354
 - symbol, 2-261
- User Runtime Library, 2-156
- username of a process, 2-265
- UTC, 2-247, 2-400
- ?UTID offset, 2-777
- :UTIL:SYSTEM_CALL_SAMPLES
 - subdirectory, A-1
- ?UTLEN value, 2-777
- ?UTPRI offset, 2-777
- ?UTSK offset, 2-156
- ?UTSTAT offset, 2-777
- ?UUID offset, 2-777

V

- ?VALAD system call, 2-781
- ?VALIDATE system call, 2-782
- validating
 - area for Read or Write access, 2-782
 - logical address, 2-781
- ?VBITM value, 2-376
- ?VCUST system call, 2-786
- ?VDELIM offset, 2-783
- verifying a customer, 2-786, 2-789

?VERRIGN value, 2-793, 2-802
 ?VERRNTR value, 2-793, 2-802
 ?VERROR offset, 2-783
 ?VERRTRM value, 2-793, 2-802
 Vertical Form Unit, loading in the VFU printer,
 2-527
 vertical format unit (VFU), 2-600-2-601
 ?VFUNC offset, 2-783
 Viewing, recent messages, in :SYSLOG and
 :CON0_LOG, 2-741
 ?VINIREV value, 2-792
 virtual PID, 2-266
 Virtual Timer Facility
 attaching, 2-149, 2-151
 cascading and setting, 2-794
 creating, 2-790
 exiting from, 2-806
 killing, 2-797
 modifying, 2-799
 restarting and suspending, 2-804
 synchronizing, 2-795
 waiting for a signal from, 2-848
 waiting for an error message from, 2-846
 ?VLAC value, 2-781
 ?VLENGTH offset, 2-783
 ?VMEM system call, 2-787-2-788
 ?VMLTH value, 2-788
 ?VMODILV value, 2-793, 2-802
 ?VMODNUL value, 2-793, 2-802
 ?VMODSIG value, 2-793, 2-802
 ?VMSTAT offset, 2-788
 volume identifier, checking, 2-35
 ?VPLTH value, 2-783
 ?VPOINTER offset, 2-783
 ?VRCUST system call, 2-789
 ?VRES offset, 2-783
 ?VRESD offset, 2-783
 ?VRING offset, 2-783
 ?VRNBR value, 2-783
 ?VSPIDS value, 2-518
 ?VTERIOVR value, 2-847
 ?VTERSOVR value, 2-847
 ?VTFCAS offset, 2-791-2-792, 2-800-2-801

?VTFCREATE system call, **2-790-2-796**,
 2-797, 2-799, 2-805, 2-847
 ?VTFENTR offset, 2-791-2-792, 2-800-2-801
 ?VTFHUNG value, 2-847
 ?VTFID offset, 2-791-2-792, 2-798,
 2-800-2-801
 ?VTFIMSK offset, 2-791, 2-794, 2-800, 2-803
 ?VTFINISUS value, 2-793
 ?VTFKILL system call, 2-797-2-798
 ?VTFKLEN value, 2-798
 ?VTFKREV value, 2-798
 ?VTFMODE offset, 2-791, 2-793, 2-800-2-803,
 2-805
 ?VTFMODIFY system call, 2-790, **2-799-2-803**
 ?VTFMODSUS value, 2-800-2-803
 ?VTFMPLN value, 2-791, 2-800
 ?VTFMREV value, 2-801
 ?VTFONESHOT value, 2-793, 2-802
 ?VTFPID offset, 2-791-2-792, 2-800-2-801
 ?VTFPRI offset, 2-791-2-792, 2-800-2-801
 ?VTFREV offset, 2-791-2-792, 2-798,
 2-800-2-801
 ?VTFRST value, 2-805
 ?VTFSAVE value, 2-793, 2-802
 ?VTFSEC value, 2-794, 2-803
 ?VTFSET offset, 2-791-2-792, 2-800-2-801
 ?VTFSKEW offset, 2-791-2-792, 2-800-2-801
 ?VTFSUD value, 2-805
 ?VTFSUS system call, 2-793, 2-803,
2-804-2-805
 ?VTFSYNC offset, 2-791-2-792, 2-800-2-801
 ?VTFRTICK value, 2-794, 2-803
 ?VTFRTID offset, 2-791-2-792, 2-800-2-801
 ?VTFUNIT offset, 2-791, 2-794, 2-800, 2-803
 ?VTFUSEC value, 2-794, 2-803
 ?VTFXIT system call, 2-806
 ?VWSMAX offset, 2-788

W

WACK character, 2-677
 waiting
 task or process, 2-845
 terminal interrupt, 2-335

?WALKBACK system call, 1–2, **2-807-2-808**

walking back through stacks, 2-807

?WDELAY system call, 2-809
examples of, A-4, A-7, A-37

?WHIST system call, 2-810-2-812

?WIN_BORDER_TYPE... values, 2-822, 2-829,
2-831

?WIN_CRE... offsets and values, 2-818-2-860

?WIN_CREATE_WINDOW function, 2-814

?WIN_DEFINE_PORTS function, 2-814, 2-823

?WIN_DELETE_WINDOW function, 2-229,
2-814, **2-822**

?WIN_DEVICE_STATUS function, 2-815,
2-837-2-838

?WIN_DEVSTAT... offsets and values,
2-838-2-839

?WIN_DISABLE_KEYBOARD function, 2-815,
2-827

?WIN_DPORT... offsets and values,
2-823-2-860

?WIN_ENABLE_KEYBOARD function, 2-815

?WIN_GET_TITLE function, 2-815, **2-832**

?WIN_GET_USER_INTERFACE function,
2-815, **2-830**

?WIN_GET_WINDOW_ID function, 2-815,
2-837

?WIN_GET_WINDOW_NAME function, 2-815,
2-832

?WIN_GINT... offsets and values, 2-830-2-860

?WIN_GTITLE... offsets and values, 2-832

?WIN_HIDE_GROUP function, 2-814, **2-826**

?WIN_HIDE_WINDOW function, 2-814, **2-826**

?WIN_LANG_ID... values, 2-839

?WIN_OUTBACK_GROUP function, 2-814,
2-825

?WIN_OUTBACK_WINDOW function, 2-814,
2-825

?WIN_PALETTE_TYPE... values, 2-821-2-822,
2-837

?WIN_PERMANENCE_OFF function, 2-815,
2-827

?WIN_PERMANENCE_ON function, 2-815,
2-827

?WIN_PKT.CHAN_NUM offset, 2-816-2-817

?WIN_PKT.FLAGS... offset and values,
2-816-2-817

?WIN_PKT.FUNC offset, 2-813, 2-816

?WIN_PKT.LEN value, 2-816

?WIN_PKT.PATH... offsets, 2-816, 2-832

?WIN_PKT.PKT_ID offset, 2-816

?WIN_PKT.SUBPKT offset, **2-816-2-817**,
2-822, 2-825-2-833, 2-837, 2-838

?WIN_PKT.WIND_ID offset, 2-816, 2-838

?WIN_PTR_TYPE... values, 2-839

?WIN_RETURN_DEVICE_WINDOWS
function, 2-815, **2-840**

?WIN_RETURN_GROUP_WINDOWS function,
2-815
preface, 2-840

?WIN_RTN_DEVWINDS... offsets and values,
2-841

?WIN_RTN_GRPWINDS... offsets and values,
2-840

?WIN_SET_TITLE function, 2-815, **2-831**

?WIN_SET_USER_INTERFACE function,
2-815, **2-828-2-860**

?WIN_SINT... offsets and values, 2-828-2-829

?WIN_STATUS... offsets and values,
2-834-2-835

?WIN_STITLE... offsets and values, 2-831

?WIN_SUSPEND_GROUP function, 2-814,
2-827

?WIN_SUSPEND_WINDOW function, 2-814,
2-826

?WIN_UNHIDE_GROUP function, 2-814,
2-826

?WIN_UNHIDE_WINDOW function, 2-814,
2-826

?WIN_UNsuspend_GROUP function, 2-815,
2-826-2-827

?WIN_UNsuspend_WINDOW function,
2-814, **2-827**

?WIN_UPFRONT_GROUP function, 2-814,
2-825

?WIN_UPFRONT_WINDOW function, 2-814,
2-825

?WIN_VT_TYPE... values, 2-836

?WIN_WINDOW_STATUS function, 2-815,
2-833-2-834

window
active group of, 2-574
graphics, 2-226
ID number, 2-226, 2-569
manipulating, 2-813

- pathname, 2-226, 2-569
- upfront, 2-574
- ?WINDOW functions
 - bringing a window or group to the front, 2-825
 - changing the view and scan ports, 2-823-2-860
 - controlling
 - keyboard input to a window, 2-827
 - window output, priority, and visibility, 2-825-2-826
 - creating new, 2-818
 - deleting, 2-229, **2-822**
 - getting
 - current user interface settings, 2-829-2-830
 - status, 2-833-2-860
 - status of a physical device, 2-837-2-860
 - title, 2-831-2-860
 - window IDs, 2-837-2-860
 - making hidden window or group visible, 2-826-2-860
 - resuming output to a suspended window, 2-827
 - sending a window or group to the back, 2-825
 - setting window
 - permanence, 2-827
 - title, 2-831-2-860
 - user interface, 2-828-2-860
 - suspending window output, 2-826
- ?WINDOW system call, 2-191, **2-813-2-841**
 - examples of, A-46-A-57
- windowing, system calls (names of), 2-9
- ?WIRE system call, 2-842-2-843
- wiring
 - Agent, 2-18
 - pages, 2-76
 - pages to the working set, 2-842
- word, task status, 2-282
- .WORD assembly language statement, 1-4
- working directory, changing the, 2-89
- working set, wiring pages to the, 2-842
- ?WRAC value, 2-781
- ?WRB system call, 2-213, **2-596-2-597**, 2-844
- ?WRITE system call, **2-604**, 2-844
 - examples of, A-3-A-4, A-11, A-14-A-16, A-19, A-22-A-23, A-26, A-29, A-34
- WRITE.SR sample program, A-2, **A-22-A-23**
- write/read access to memory, 2-782
- writing
 - block I/O, 2-596, 2-844

- record I/O, 2-604
 - user data area, 2-602-2-604, 2-844
- ?WRUDA system call, 2-213, **2-602**, 2-844
- WSB, WSL, and WSP stack registers, 2-314
- ?WTSIG system call, 2-689, 2-691, **2-845**, 2-848
- ?WTVERR system call, 2-793, 2-802, **2-846**
- ?WTVSIG system call, 2-791, **2-848**
- ?WVBPL value, 2-783
- ?WVWPL value, 2-783
- ?XWM1 offsets, **2-135**

X

- ?X0FC offset, 2-198
- ?X1FC offset, 2-198
- ?X1LTH value, 2-113
- ?X2AP value, 2-121
- ?X2CD value, 2-120
- ?X2CM value, 2-121
- ?X2CN value, 2-120
- ?X2CP value, 2-121
- ?X2DD value, 2-121
- ?X2EB value, 2-120
- ?X2EX value, 2-120
- ?X2HO value, 2-120
- ?X2LN value, 2-120
- ?X2RC value, 2-121
- ?X2RM value, 2-121
- ?X2SD value, 2-121
- ?X2SE value, 2-120
- ?X2TO value, 2-121
- ?X3CO value, 2-121
- ?X3TR value, 2-121
- ?XAFD offset, 2-113, 2-118, 2-131, 2-133, 2-135
- ?XAFT offset, 2-113, 2-118, 2-131, 2-133, 2-135
- ?XBFXR value, So String, 2-894
- ?XBIDS value, 2-894
- ?XBNHR value, 2-894
- ?XBORD value, 2-894
- ?XBTRP value, 2-894

?XCREA_DIR... offsets, 2-855-2-856
 ?XCREA_IPC... offsets, 2-856-2-857
 ?XCREA_LNK... offsets, 2-857
 ?XCREA_OTH... offsets, 2-854-2-855
 ?XCREA_PKT... offsets, 2-850-2-851
 ?XCREA_TIME... offsets, 2-853
 ?XCREATE system call, 2-849-2-860
 ?XD2FG offset, 2-138-2-158
 ?XD3FG offset, 2-138-2-158
 ?XDAD offset, 2-139-2-140
 ?XDAT offset, 2-113, 2-114, 2-131, 2-133
 ?XDDBP offset, 2-139-2-140
 ?XD COP offset, 2-139-2-140
 ?XD EP offset, 2-139-2-140
 ?XD EV offset, 2-110-2-111
 ?XD JN offset, 2-139-2-140
 ?XD LMT offset, 2-139-2-140
 ?XD PN offset, 2-139-2-140
 ?XD QP offset, 2-139-2-140
 ?XD RSV offset, 2-139-2-140
 ?XD SD offset, 2-139-2-140
 ?XD SFG offset, 2-138-2-158
 ?XD SQN offset, 2-139-2-140
 ?XD ST offset, 2-139-2-140
 ?XD TA offset, 2-139-2-140
 ?XD UL offset, 2-109
 ?XD UN offset, 2-139-2-140
 ?XD UT offset, 2-109
 XEQ DEBUG command, 2-255
 ?XFBAT queue type, 2-114
 ?XFBI value, 2-117, 2-125
 ?XFBP offset, 2-113, 2-118, 2-131, 2-133,
 2-135
 ?XFDA value, 2-116, 2-125
 ?XFDUN function, 2-109
 ?XFEP value, 2-115
 ?XFFO value, 2-117, 2-126
 ?XFFTA queue type, 2-114
 ?XFG2 offset, 2-113, 2-115, 2-131, 2-133,
 2-138, 2-139-2-140
 ?XFGRT value, 2-894
 ?XFGS offset, 2-113, **2-116**, 2-117, 2-122,
 2-123, 2-131, 2-133, 2-138-2-158
 ?XFGWD value, 2-894
 ?XFHAM queue type, 2-114
 ?XFHK offset, 2-136, 2-138, 2-139
 ?XF LC value, 2-110-2-111
 ?XF LK offset, 2-136, 2-138, 2-139
 ?XF LO value, 2-110-2-111
 ?XF LPT queue type, 2-114
 ?XF ME value, 2-110-2-111
 ?XF MLT function, 2-107-2-108, 2-131
 ?XF MNT function, **2-131**
 ?XF MOD function, **2-133**, 2-134, 2-471
 ?XF MUN function, 2-106, 2-131
 ?XF NH and ?XF8B values, 2-115
 ?XF NO value, 2-117, 2-125
 ?XF NQN function, 2-136-2-137
 ?XF NR value, 2-116, 2-125
 ?XF NV value, 2-110-2-111
 ?XF OP value, 2-117, 2-125
 ?XF OTH queue type, 2-114
 ?XF P1 and ?XF P2 offsets, 2-127-2-131
 ?XF P2L offset, 2-127-2-131
 ?XF P3 and ?XF P4 offsets, 2-130-2-158
 ?XF PE value, 2-116, 2-126, 2-131
 ?XF PLT queue type, 2-114
 ?XF QDS function, 2-138
 ?XF QN offset, 2-138, 2-139
 ?XF RA value, 2-117, 2-126
 ?XF SH value, 2-116, 2-117, 2-125
 ?XF SNA queue type, 2-114
 ?XF STAT file types, 2-863
 ?XF STAT system call, 2-862
 ?XF STAT_PKT... offsets, 2-866-2-876
 ?XF STS function, 2-129-2-130
 ?XF SUB queue type, 2-112, 2-114, 2-131
 ?XF TB value, 2-115
 ?XF TI value, 2-117-2-118, 2-126
 ?XF UC value, 2-117
 ?XF WP offset, 2-136, 2-139
 ?XF XDU function, 2-132
 ?XF XML function, 2-108-2-109

?XFXTS function, 2-130
 ?XFXUN function, 2-107, 2-108
 ?XGTACP system call, 2-882
 ?XGTACP_... offsets, 2-883-2-884
 ?XHBP offset, 2-113, 2-122, 2-133
 ?XICNT offset, **2-893**, 2-895
 ?XID1-?XID3 offsets, **2-893**, 2-895
 ?XIDIR offset, **2-893**, 2-895
 ?XIFUN offset, 2-893-2-894
 ?XIGTD value, 2-894
 ?XII1 and ?XII2 values, 2-894
 ?XILDN offset, **2-893**, 2-895
 ?XILN value, 2-893
 ?XINIT system call, 2-886
 ?XINIT_CACHE value, 2-889
 ?XINIT_FIXUP_REC value, 2-890
 ?XINIT_LDID_SPECIFIED value, 2-889
 ?XINIT_NO_HARDWARE value, 2-889
 ?XINIT_OVERRIDE value, 2-889
 ?XINIT_PKT... offsets, 2-888-2-910
 ?XINIT_PUL_PKTID value, 2-892
 ?XINIT_ROOT value, 2-889
 ?XINIT_TARGET_DIR value, 2-889
 ?XINIT_TRESPASS value, 2-889
 ?XINIT_WORKING_DIR value, 2-889
 ?XIOP offset, 2-893-2-894
 ?XIP1-?XIP3 offsets, **2-893-2-910**
 ?XIPHI offset, 2-893-2-894
 ?XIPIO offset, 2-893-2-894
 ?XIR1-?XIR9 offsets, 2-893-2-910
 ?XIRA offset, **2-893**, 2-896
 ?XIRES offset, 2-893-2-894
 ?XLCAN value, 2-127-2-131
 ?XLDUN value, 2-109
 ?XLFWP value, 2-139-2-140
 ?XLHOL value, 2-127-2-131
 ?XLLC value, 2-110
 ?XLLO value, 2-110
 ?XLME value, 2-110
 ?XLMLT value, 2-107-2-108
 ?XLMNT value, 2-131
 ?XLMOD value, 2-133
 ?XLMT offset, 2-113, 2-115, 2-116, 2-131,
 2-133, 2-135
 ?XLMUN value, 2-106
 ?XLNQN value, 2-137
 ?XLNV value, 2-110
 ?XLPN offset, 2-113, 2-115
 ?XLQDS value, 2-139
 ?XLQNB value, 2-137
 ?XLSTS value, 2-129
 ?XLTH value, 2-113
 ?XLUNH value, 2-127-2-131
 ?XLXTS value, 2-130
 ?XMBP offset, 2-113, 2-122, 2-133
 ?XMDF offset, 2-110-2-111
 ?XMEL value, 2-108
 ?XMFC value, 2-108
 ?XMFG offset, 2-110-2-111
 ?XMFI value, 2-108
 ?XMFN offset, 2-110-2-111
 ?XMFR value, 2-108
 ?XMFS value, 2-111
 ?XMIBM value, 2-35
 ?XMLE offset, 2-108
 ?XMLF offset, 2-108
 ?XMLL offset, 2-107-2-108
 ?XMLR offset, 2-108
 ?XMLS offset, 2-108
 ?XMLT offset, 2-107-2-108
 ?XMLV offset, 2-107-2-108
 ?XMLX value, 2-108
 ?XMSQ offset, 2-110-2-111
 ?XMSQB value, 2-111
 ?XMT system call, 2-898
 ?XMTW system call, 2-899
 example of, A-33
 ?XMUE offset, 2-107
 ?XMUF offset, 2-107
 ?XMUL offset, 2-106
 ?XMUQ offset, 2-107
 ?XMUR offset, 2-107
 ?XMUS offset, 2-107

?XMUT offset, 2-106, 2-109
 ?XMUX value, 2-107
 ?XNRQ offset, 2-139
 ?XNRT offset, 2-139
 XON value, 2-41
 ?XPBP offset, 2-113, 2-118, 2-131
 ?XPCH offset, 2-902, 2-906
 ?XPCID offset, 2-903, 2-907
 ?XPCPL offset, 2-902, 2-906
 ?XPCPU offset, 2-903, 2-907
 ?XPCW offset, 2-902, 2-906
 ?XPDBS offset, 2-902, 2-907
 ?XPDESLN offset, 2-904, 2-907
 ?XPDIS offset, 2-902, 2-907, 2-909
 ?XPDRS offset, 2-902, 2-907
 ?XPEX offset, 2-902, 2-906
 ?XPF2-?XPF5 offsets, 2-902, 2-905-2-906
 ?XPFA offset, 2-902, 2-907
 ?XPFL offset, 2-902, 2-905
 ?XPFP offset, 2-902, 2-905
 ?XPGAB offset, 2-909
 ?XPGBS offset, 2-903, 2-907
 ?XPGLT offset, 2-909
 ?XPGRB offset, 2-909
 ?XPGRS offset, 2-903, 2-907
 ?XPIH offset, 2-902, 2-907
 ?XPLFA offset, 2-903, 2-907
 ?XPLL offset, 2-903, 2-907
 ?XPLTH offset, 2-909
 ?XPLTH value, 2-903
 ?XPMX offset, 2-902, 2-906
 ?XPNBS offset, 2-903, 2-908
 ?XPNR offset, 2-902, 2-905
 ?XPNRS offset, 2-903, 2-908
 ?XPPD offset, 2-902, 2-906
 ?XPPG offset, 2-903, 2-907
 ?XPPH offset, 2-902, 2-906
 ?XPPL offset, 2-903, 2-907
 ?XPPR offset, 2-902, 2-906
 ?XPPU offset, 2-903, 2-908
 ?XPPV offset, 2-902, 2-906
 ?XPR1 offset, 2-902, 2-906
 ?XPRH offset, 2-902, 2-906
 ?XPRI offset, 2-113, 2-131, 2-133, 2-135
 ?XPRI value, 2-116
 ?XPRV offset, 2-105-2-112, 2-113, 2-114
 ?XPSF offset, 2-902, 2-905
 ?XPSID value, 2-905
 ?XPSID1 and ?XPSID2 values, 2-905, 2-909
 ?XPSL offset, 2-903, 2-907
 ?XPSNS offset, 2-902, 2-905
 ?XPSP offset, 2-902, 2-905, 2-908
 ?XPSQF offset, 2-902, 2-905
 ?XPSTAT system call, 2-900-2-909
 ?XPSW offset, 2-902, 2-905
 ?XPSWS offset, 2-904
 ?XPUBS offset, 2-903, 2-908
 ?XPULC offset, 2-903, 2-907
 ?XPUN offset, 2-903, 2-908
 ?XPUPD offset, 2-903, 2-908
 ?XPUQSH offset, 2-909
 ?XPURS offset, 2-903, 2-908
 ?XPUWS offset, 2-904
 ?XPWD offset, 2-113, 2-122, 2-133
 ?XPWM offset, 2-902, 2-906
 ?XPWS offset, 2-902, 2-906
 ?XQ1FG offset, 2-137
 ?XQNJ offset, 2-137
 ?XQQN offset, 2-137
 ?XQQT offset, 2-137
 ?XQRSV offset, 2-137
 ?XRES1-?XRES4 offsets, 2-110-2-112, 2-122,
 2-124, 2-131, 2-133
 ?XRFNC function, 2-132
 ?XRFNC offset, 2-105-2-112
 ?XSCV value, 2-111
 ?XSD1 and ?XSD2 values, 2-111
 ?XSDEN value, 2-111
 ?XSEL value, 2-111
 ?XSEQ offset, 2-113, 2-118, 2-131, 2-133,
 2-134
 ?XSIBM value, 2-111
 ?XSMT value, 2-111

?XSRO value, 2-111
 ?XSVU value, 2-111
 ?XT16T value, 2-304.2, 2-762
 ?XT32T value, 2-304.2, 2-762
 ?XTABR value, 2-304.2, 2-762
 ?XTAOS value, 2-304.2, 2-762
 ?XTBCX value, 2-304.2, 2-762
 ?XTCCX value, 2-304.2, 2-762
 ?XTCIN value, 2-304.2, 2-762
 ?XTIM offset, 2-113, 2-115, 2-131, 2-133
 ?XTR16 value, 2-304.2, 2-762
 ?XTR32 value, 2-304.2, 2-762
 ?XTSUP value, 2-304.2, 2-762
 ?XTYP offset, 2-113, 2-114, 2-131, 2-133,
 2-134
 ?XUSR offset, 2-113, 2-122, 2-133
 XVCT instruction, 2-155
 ?XVOL offset, 2-110-2-111
 XW0-XW3 values, 2-133
 ?XWM0-?XWM3 offsets, 2-133, **2-135**
 ?XWW0-?XWW3 offsets, 2-131, 2-133, **2-135**
 ?XWW0L-?XWW3L offsets, 2-131, 2-133
 ?XWW1 offset, 2-135
 ?XWW2 offset, 2-135
 ?XWW3 offset, 2-135
 ?XXDL offset, 2-132
 ?XXDP offset, 2-132
 ?XXDQ offset, 2-132
 ?XXDT offset, 2-132
 ?XXW0-?XXW3 offsets, 2-113, 2-119-2-120
 ?XXW0L-?XXW3L offsets, 2-113, 2-119-2-120

Y

year, current, 2-187

Z

?ZAC function, 2-439
 ?ZAC3 and ?ZAC4 offsets, 2-442
 ?ZACA offset, 2-442
 ?ZACF offset, 2-442
 ?ZACI offset, 2-442
 ?ZACR offset, 2-442
 ?ZACZ value, 2-442
 ?ZAG function, 2-439
 ?ZAG value, 2-443
 ?ZAGF offset, 2-443
 ?ZAGI offset, 2-443
 ?ZAGL value, 2-443
 ?ZAGN offset, 2-443
 ?ZAGZ value, 2-443
 ?ZAK value, 2-441
 ?ZAL function, 2-439
 ?ZBI function, 2-439
 ?ZBIA offset, 2-446
 ?ZBIB offset, 2-446
 ?ZBIF offset, 2-446
 ?ZBII offset, 2-446
 ?ZBIL value, 2-446
 ?ZBIZ value, 2-446
 ?ZBL function, 2-439
 ?ZBLA offset, 2-444
 ?ZBLB offset, 2-444
 ?ZBLF offset, 2-444
 ?ZBLI offset, 2-444
 ?ZBLL value, 2-444
 ?ZBLZ value, 2-444
 ?ZBO function, 2-439
 ?ZBOA offset, 2-445
 ?ZBOB offset, 2-445
 ?ZBOF offset, 2-445
 ?ZBOI offset, 2-445
 ?ZBOL value, 2-445
 ?ZBOZ value, 2-445
 ?ZBR function, 2-439
 ?ZBRF offset, 2-447
 ?ZBRI offset, 2-447
 ?ZBRL value, 2-447
 ?ZBRS offset, 2-447
 ?ZBRZ value, 2-447
 ?ZCA function, 2-439
 ?ZCAI offset, 2-448

?ZCAL value, 2-448
 ?ZCAR offset, 2-448
 ?ZCAS offset, 2-448
 ?ZCAZ value, 2-448
 ?ZCL function, 2-439
 ?ZCO function, 2-439
 ?ZCOF offset, 2-451
 ?ZCOI offset, 2-451
 ?ZCOL value, 2-451
 ?ZCOS offset, 2-451
 ?ZCP function, 2-439
 ?ZCPI offset, 2-452
 ?ZCPL value, 2-452
 ?ZCPN offset, 2-452
 ?ZCPR offset, 2-452
 ?ZCPZ value, 2-452
 ?ZCR function, 2-439, 2-451, 2-453, 2-477
 ?ZCRF offset, 2-453
 ?ZCRI offset, 2-453
 ?ZCRL value, 2-453
 ?ZCRN offset, 2-453
 ?ZCRQ offset, 2-453
 ?ZCRR offset, 2-453
 ?ZCRZ value, 2-453
 ?ZCS function, 2-439
 ?ZCS3 offset, 2-449-2-450
 ?ZCSA offset, 2-449-2-450
 ?ZCSB offset, 2-449-2-450
 ?ZCSF offset, 2-449-2-450
 ?ZCSI offset, 2-449-2-450
 ?ZCSK offset, 2-449-2-450
 ?ZCSL value, 2-449-2-450
 ?ZCSN offset, 2-449-2-450
 ?ZCSP offset, 2-449-2-450
 ?ZCSR offset, 2-449-2-450
 ?ZCSU offset, 2-449-2-450
 ?ZCSZ value, 2-450
 ?ZDE function, 2-439
 ?ZDF function, 2-439
 ?ZDFA offset, 2-454
 ?ZDFB offset, 2-454
 ?ZDFE offset, 2-454
 ?ZDFI offset, 2-454
 ?ZDFL value, 2-454
 ?ZDFZ value, 2-454
 ?ZDI function, 2-439
 ?ZDIF offset, 2-455
 ?ZDII offset, 2-455
 ?ZDIL value, 2-455
 ?ZDIN offset, 2-455
 ?ZDIZ value, 2-455
 ?ZDM function, 2-439
 ?ZDM value, 2-471
 ?ZDS function, 2-439
 ?ZDSF offset, 2-456
 ?ZDSI offset, 2-456
 ?ZDSL value, 2-456
 ?ZDSM offset, 2-456
 ?ZDSZ value, 2-456
 ?ZEL function, 2-439
 ?ZELF offset, 2-457
 ?ZELI offset, 2-457
 ?ZELN value, 2-457
 ?ZELR offset, 2-457
 ?ZELZ value, 2-457
 ?ZEN function, 2-439
 ?ZEN3 offset, 2-458-2-459
 ?ZENC offset, 2-458-2-459
 ?ZENF offset, 2-458-2-459
 ?ZENI offset, 2-458-2-459
 ?ZENK offset, 2-458-2-459
 ?ZENL value, 2-458
 ?ZENR offset, 2-458-2-459
 ?ZENT offset, 2-458-2-459
 ?ZENZ value, 2-459
 ?ZEV function, 2-439
 ?ZEVF offset, 2-460
 ?ZEVI offset, 2-460
 ?ZEVL value, 2-460
 ?ZEVZ value, 2-460
 ?ZFL function, 2-439

?ZFLF offset, 2-461
?ZFLI offset, 2-461
?ZFLL value, 2-461
?ZFLLS offset, 2-461
?ZFLZ value, 2-461
?ZFN function, 2-439
?ZFO function, 2-439
?ZFOA offset, 2-462
?ZFOB offset, 2-462
?ZFOF offset, 2-462
?ZFOI offset, 2-462
?ZFOL value, 2-462
?ZFOZ value, 2-462
?ZHA function, 2-439
?ZHA value, 2-441
?ZHAF offset, 2-463
?ZHAI offset, 2-463
?ZHAL value, 2-463
?ZHAR offset, 2-463
?ZHAZ value, 2-463
?ZHE function, 2-439
?ZHEH offset, 2-464
?ZHEI offset, 2-464
?ZHEL value, 2-464
?ZHER offset, 2-464
?ZHEZ value, 2-464
?ZHO function, 2-439
?ZHOI offset, 2-465
?ZHOL value, 2-465
?ZHOR offset, 2-465
?ZHOS offset, 2-465
?ZHOZ value, 2-465
?ZJ30 value, 2-476
?ZJ90 value, 2-480
?ZJF1 value, 2-446
?ZJH0 value, 2-457
?ZJI0 value, 2-460
?ZJS1 value, 2-482
?ZJV1-?ZJV9 values, 2-496
?ZJVD-?ZJVE values, 2-496
?ZLI function, 2-439

?ZLIA offset, 2-466
?ZLIB offset, 2-466
?ZLII offset, 2-466
?ZLIL value, 2-466
?ZLIZ value, 2-466
?ZLO function, 2-439
?ZLO3-?ZLO6 offsets, 2-467-2-468
?ZLOC offset, 2-467-2-468
?ZLOF offset, 2-467-2-468
?ZLOI offset, 2-467-2-468
?ZLOL value, 2-467
?ZLOP offset, 2-467-2-468
?ZLOZ value, 2-468
?ZLP function, 2-439
?ZLPI offset, 2-469
?ZLPL value, 2-469
?ZLPN offset, 2-469
?ZLPR offset, 2-469
?ZLPZ value, 2-469
?ZMD function, 2-439
?ZMD value, 2-441
?ZME function, 2-439
?ZMO function, 2-439
?ZMOA offset, 2-472
?ZMOB offset, 2-472
?ZMOF offset, 2-472
?ZMOI offset, 2-472
?ZMOL value, 2-472
?ZMOM offset, 2-472
?ZMOR offset, 2-472
?ZMOZ value, 2-472
?ZMP function, 2-439
?ZMPA offset, 2-470
?ZMPB offset, 2-470
?ZMPF offset, 2-470
?ZMPI offset, 2-470
?ZMPL value, 2-470
?ZMPZ value, 2-470
?ZMS function, 2-439
?ZMS0 offset, 2-473
?ZMS3-?ZMS9 offsets, 2-473-2-474

?ZMSA offset, 2-473
 ?ZMSD offset, 2-473
 ?ZMSE offset, 2-473-2-474
 ?ZMSF offset, 2-473-2-474
 ?ZMSG offset, 2-473-2-474
 ?ZMSI offset, 2-473-2-474
 ?ZMSK offset, 2-473-2-474
 ?ZMSL value, 2-473
 ?ZMSM offset, 2-473
 ?ZMSP offset, 2-473-2-474
 ?ZMSQ value, 2-474
 ?ZMSR offset, 2-473
 ?ZMSS offset, 2-473
 ?ZMST offset, 2-473
 ?ZMSU offset, 2-473
 ?ZMSV offset, 2-473
 ?ZMSX offset, 2-473
 ?ZMSZ offset, 2-473
 ?ZON function, 2-439
 ?ZOP function, 2-439
 ?ZOPF offset, 2-476
 ?ZOPI offset, 2-476
 ?ZOPL value, 2-476
 ?ZOPR offset, 2-476
 ?ZOPZ value, 2-476
 ?ZPA function, 2-439
 ?ZPAF offset, 2-477
 ?ZPAI offset, 2-477
 ?ZPAL value, 2-477
 ?ZPAS offset, 2-477
 ?ZPAZ value, 2-477
 ?ZPE function, 2-440
 ?ZPE3 and ?ZPE4 offsets, 2-478
 ?ZPEF offset, 2-478
 ?ZPEI offset, 2-478
 ?ZPEL value, 2-478
 ?ZPER offset, 2-478
 ?ZPEU offset, 2-478
 ?ZPEV offset, 2-478
 ?ZPEZ value, 2-478
 ?ZPI function, 2-440
 ?ZPIA offset, 2-479
 ?ZPIB offset, 2-479
 ?ZPIF offset, 2-479
 ?ZPII offset, 2-479
 ?ZPIL value, 2-479
 ?ZPIR offset, 2-479
 ?ZPIZ value, 2-479
 ?ZPR function, 2-440
 ?ZPR value, 2-441
 ?ZPRF offset, 2-480
 ?ZPRI offset, 2-480
 ?ZPRL value, 2-480
 ?ZPRR offset, 2-480
 ?ZPRZ value, 2-480
 ?ZPU function, 2-440
 ?ZQP function, 2-440
 ?ZQPA offset, 2-481-2-482
 ?ZQPB offset, 2-481-2-482
 ?ZQPC offset, 2-481-2-482
 ?ZQPF offset, 2-481-2-482
 ?ZQPH offset, 2-481
 ?ZQPI offset, 2-481-2-482
 ?ZQPL value, 2-481
 ?ZQPN offset, 2-481
 ?ZQPO offset, 2-481
 ?ZQPP offset, 2-481
 ?ZQPS offset, 2-481
 ?ZQPU offset, 2-481
 ?ZQPX value, 2-481
 ?ZQPZ value, 2-482
 ?ZRE function, 2-440
 ?ZREF offset, 2-484
 ?ZREI offset, 2-484
 ?ZREL value, 2-484
 ?ZRER offset, 2-484
 ?ZREZ value, 2-484
 ?ZRF function, 2-440
 ?ZRF value, 2-441
 ?ZRFF offset, 2-483
 ?ZRFI offset, 2-483
 ?ZRFL value, 2-483

?ZRFM offset, 2-483
 ?ZRFZ value, 2-483
 ?ZRT function, 2-440
 ?ZRTA offset, 2-485
 ?ZRTB offset, 2-485
 ?ZRTI offset, 2-485
 ?ZRTL value, 2-485
 ?ZRTZ value, 2-485
 ?ZS1B offset, 2-487-2-490
 ?ZS1C offset, 2-487-2-490
 ?ZS1F offset, 2-487-2-490
 ?ZS1L value, 2-487
 ?ZS1M offset, 2-487-2-490
 ?ZS1P offset, 2-487-2-490
 ?ZS1S offset, 2-487-2-490
 ?ZSI function, 2-440
 ?ZSIF offset, 2-486
 ?ZSII offset, 2-486
 ?ZSIL value, 2-486
 ?ZSIS offset, 2-486
 ?ZSIZ value, 2-486
 ?ZSK function, 2-440
 ?ZSK3 offset, 2-490
 ?ZSKF offset, 2-490
 ?ZSKI offset, 2-490
 ?ZSKL value, 2-490
 ?ZSKR offset, 2-490
 ?ZSKZ value, 2-491
 ?ZSP function, 2-440
 ?ZSP3-?ZSP9 offsets, 2-487-2-488
 ?ZSPB offset, 2-487-2-488
 ?ZSPC offset, 2-487-2-488
 ?ZSPD offset, 2-487-2-488
 ?ZSPE offset, 2-487-2-490
 ?ZSPF offset, 2-487-2-488
 ?ZSPH offset, 2-487-2-488
 ?ZSPI offset, 2-487-2-488
 ?ZSPK offset, 2-487-2-488
 ?ZSPN offset, 2-487-2-488
 ?ZSPP offset, 2-487-2-488
 ?ZSPQ offset, 2-487-2-488
 ?ZSPR offset, 2-487-2-490
 ?ZSPS offset, 2-487-2-488
 ?ZSPT offset, 2-487-2-488
 ?ZSPV offset, 2-487-2-488
 ?ZSPZ value, 2-488
 ?ZSR function, 2-440, **2-492**
 ?ZSR3-?ZSR7 offsets, 2-492
 ?ZSRB offset, 2-492
 ?ZSRC offset, 2-492
 ?ZSRF offset, 2-492
 ?ZSRI offset, 2-492
 ?ZSRL value, 2-492
 ?ZSRM offset, 2-492
 ?ZSRQ offset, 2-492
 ?ZSRR offset, 2-492
 ?ZSRS offset, 2-492
 ?ZSRX offset, 2-492
 ?ZSRZ value, 2-493
 ?ZSS function, 2-440
 ?ZSS0-?ZSS9 offsets, 2-495, 2-497-2-498
 ?ZSSA-?ZSSZ offsets, 2-495-2-498
 ?ZST function, 2-440
 ?ZSTA offset, 2-499
 ?ZSTB offset, 2-499
 ?ZSTF offset, 2-499
 ?ZSTI offset, 2-499
 ?ZSTL value, 2-499
 ?ZSTZ value, 2-499
 ?ZSX0-?ZSX9 offsets, 2-495-2-498
 ?ZTE function, 2-440
 ?ZTR function, 2-440
 ?ZTRI offset, 2-500
 ?ZTRL value, 2-500
 ?ZTRR offset, 2-500
 ?ZTRT offset, 2-500
 ?ZTRZ value, 2-500
 ?ZUC function, 2-440
 ?ZUH function, 2-440
 ?ZUHI offset, 2-501
 ?ZUHL value, 2-501
 ?ZUHR offset, 2-501

?ZUHS offset, 2-501
 ?ZUHZ value, 2-501
 ?ZUL function, 2-440
 ?ZULF offset, 2-504
 ?ZULI offset, 2-504
 ?ZULL value, 2-504
 ?ZULS offset, 2-504
 ?ZULZ value, 2-504
 ?ZUN function, 2-440
 ?ZUNF offset, 2-505
 ?ZUNI offset, 2-505
 ?ZUNL value, 2-505
 ?ZUNS offset, 2-505
 ?ZUNZ value, 2-505
 ?ZUS function, 2-440
 ?ZUS3-?ZUS7 offsets, 2-502-2-503
 ?ZUSF offset, 2-502-2-503
 ?ZUSI offset, 2-502-2-503
 ?ZUSK offset, 2-502-2-503
 ?ZUSL value, 2-502
 ?ZUSM offset, 2-502-2-503
 ?ZUSP offset, 2-502-2-503
 ?ZUSR offset, 2-502-2-503
 ?ZUSU offset, 2-502-2-503
 ?ZUSV offset, 2-502-2-503
 ?ZUSZ value, 2-503
 ?ZVE function, 2-440
 ?ZVEF offset, 2-508
 ?ZVEI offset, 2-508
 ?ZVEL value, 2-508
 ?ZVES offset, 2-508
 ?ZVEZ value, 2-508
 ?ZXB function, 2-440
 ?ZXFG offset, 2-438, 2-441
 ?ZXFU offset, 2-438-2-440
 ?ZXID offset, 2-438, 2-439
 ?ZXIZ value, 2-439
 ?ZXLN value, 2-438
 ?ZXNA offset, 2-438, 2-441
 ?ZXNB offset, 2-438, 2-441
 ?ZXNL offset, 2-438, 2-441
 ?ZXRS offset, 2-438, 2-441
 ?ZXSP offset, 2-438, 2-441
 ?ZXTP offset, 2-438, 2-441
 ?ZY00 value, 2-456
 ?ZY10 value, 2-472
 ?ZY20-?ZY29 values, 2-474
 ?ZY2G-?ZY2J values, 2-474
 ?ZY30 value, 2-476
 ?ZY40 value, 2-478
 ?ZY50 value, 2-483
 ?ZY60 value, 2-484
 ?ZY70-?ZY74 values, 2-503
 ?ZY80-?ZY87 values, 2-468
 ?ZY90 value, 2-480
 ?ZY9G-?ZY9I values, 2-442
 ?ZYG0-?ZYG9 values, 2-450
 ?ZYC0-?ZYC2 values, 2-455
 ?ZYD0-?ZYD9 values, 2-459
 ?ZYDA and ?ZYDB values, 2-459
 ?ZYE0 value, 2-443
 ?ZYF0-?ZYF2 values, 2-446
 ?ZYG0 value, 2-454
 ?ZYH0 value, 2-457
 ?ZYI0 value, 2-460
 ?ZYJ0 value, 2-462
 ?ZYJ2 value, 2-446
 ?ZYK0-?ZYK9 values, 2-488
 ?ZYKA and ?ZYKB values, 2-488
 ?ZYL0 value, 2-444
 ?ZYM0 value, 2-445
 ?ZYMF value, 2-470
 ?ZYN0 value, 2-447
 ?ZY00 value, 2-451
 ?ZYP0 value, 2-453
 ?ZYQ0 value, 2-477
 ?ZYR0-?ZYR2 values, 2-479
 ?ZYS0 and ?ZYS1 values, 2-482
 ?ZYT0 value, 2-486
 ?ZYOU-?ZYU8 values, 2-493
 ?Zyv0-?Zyv9 values, 2-496
 ?ZYVA-?ZYVE values, 2-496

?ZYW0 value, 2-499
?ZYX0 value, 2-504
?ZYY0 value, 2-505
?ZYZ0 value, 2-508
?ZZ00 value, 2-456
?ZZ10 value, 2-472
?ZZ20-?ZZ29 values, 2-474
?ZZ2G-?ZZ2J values, 2-474
?ZZ30 value, 2-476
?ZZ40 value, 2-478
?ZZ50 value, 2-483
?ZZ60 value, 2-484
?ZZ70-?ZZ74 values, 2-503
?ZZ80-?ZZ87 values, 2-468
?ZZ90 value, 2-480
?ZZ9G-?ZZ9I values, 2-442
?ZZB0-?ZZB9 values, 2-450
?ZZC0-?ZZC2 values, 2-455
?ZZD0-?ZZD9 values, 2-459
?ZZDA and ?ZZDB values, 2-459
?ZZE0 value, 2-443
?ZZF0-?ZZF2 values, 2-446
?ZZG0 value, 2-454

?ZZH0 value, 2-457
?ZZI0 value, 2-460
?ZZJ0 value, 2-462
?ZZK0-?ZZK9 values, 2-488
?ZZKA-?ZZKC values, 2-488
?ZZL0 value, 2-444
?ZZM0 value, 2-445
?ZZMF value, 2-470
?ZZN0 value, 2-447
?ZZO0 value, 2-451
?ZZP0 value, 2-453
?ZZQ0 value, 2-477
?ZZR0-?ZZR2 values, 2-479
?ZZS0 and ?ZZS1 values, 2-482
?ZZT0 value, 2-486
?ZZU0-?ZZU8 values, 2-493
?ZZV0-?ZZV9 values, 2-496
?ZZVA-?ZZVE values, 2-496
?ZZW0 value, 2-499
?ZZX0 value, 2-504
?ZZY0 value, 2-505
?ZZZ0 value, 2-508

Document Set

For Users

AOS/VS and AOS/VS II Glossary (069-000231)

For all users, this manual defines important terms used in AOS/VS and AOS/VS II manuals, both regular and preinstalled.

Learning to Use Your AOS/VS System (069-000031)

A primer for all users, this manual introduces AOS/VS (but the material applies to AOS/VS II) through interactive sessions with the CLI, the SED and SPEED text editors, programming languages, Assembler, and the Sort/Merge utility.

Using the CLI (AOS and AOS/VS) is a good follow-up.

SED Text Editor User's Manual (AOS and AOS/VS) (093-000249)

For all users, this manual explains how to use SED, an easy-to-use screen-oriented text editor that lets you program function keys to make repetitive tasks easier. The *SED Text Editor* template (093-000361) accompanies this manual.

Using the AOS/VS System Management Interface (SMI) (069-000203)

Using the AOS/VS II System Management Interface (SMI) (069-000311)

For those working with preinstalled systems and those on regular systems who want an alternative to the CLI, the SMI is an easy-to-use, menu-driven program that helps with system management functions and some file maintenance tasks.

Using the CLI (AOS/VS and AOS/VS II) (093-000646)

For all users, this manual explains the AOS/VS and AOS/VS II file and directory structure and how to use the CLI, a command line interpreter, as the interface to the operating system. This manual explains how to use the CLI macro facility, and includes a dictionary of CLI commands and pseudomacros.

For System Managers and Operators

AOS/VS and AOS/VS II Error and Status Messages (093-000540)

For all users, but especially for system managers and operators of regular systems, this manual lists error and status messages, their source and meaning, and appropriate responses. This manual complements *Installing, Starting, and Stopping AOS/VS*; *Installing, Starting, and Stopping AOS/VS II*; and *Managing AOS/VS and AOS/VS II*.

AOS/VS and AOS/VS II Menu-Based Utilities (093-000650)

A keyboard template to identify function keys. A number of system management programs—such as Disk Jockey, VSGEN, and the SMI—and the BROWSE utility use the function keys identified on this template.

Information Update: Starting Your ECLIPSE MV/1000 DC (014–001728)

Updates *Starting and Updating Preinstalled AOS/VS* and *Starting and Updating Preinstalled AOS/VS II*.

Installing, Starting, and Stopping AOS/VS (093–000675)

Installing, Starting, and Stopping AOS/VS II (093–000539)

For system managers and operators of regular (as opposed to preinstalled) systems, these manuals explain the steps necessary to format disks, install a tailored operating system, create the multiuser environment, update the system or microcode, and routinely start up and shut down the system. *AOS/VS and AOS/VS II Error and Status Messages* and *Managing AOS/VS and AOS/VS II* are companions to these manuals.

Managing AOS/VS and AOS/VS II (093–000541)

For system managers and operators, this manual explains managing an AOS/VS or AOS/VS II system. Managing tasks include such topics as editing user profiles, managing the multiuser environment with the EXEC program, backing up and restoring files, using runtime tools, and so forth. This manual complements the “Installing” manuals, whether for regular or preinstalled systems.

Starting and Updating Preinstalled AOS/VS (069–000293)

Starting and Updating Preinstalled AOS/VS II (069–000294)

For those working with preinstalled (as opposed to regular) operating systems on all computers except ECLIPSE MV/3500™ DC and MV/5000 Series systems, these manuals explain how to start, update, and change certain system parameters. The manuals also help you interpret error messages and codes. Companion manuals are *Using the AOS/VS System Management Interface* and *Using the AOS/VS II System Management Interface*.

Starting and Updating Preinstalled AOS/VS on ECLIPSE MV/3500™ DC and MV/5000™ DC Series Systems (069–000481)

Starting and Updating Preinstalled AOS/VS II on ECLIPSE MV/3500™ DC and MV/5000™ DC Series Systems (069–000480)

For those working with preinstalled (as opposed to regular) operating systems on ECLIPSE MV/3500™ DC and MV/5000™ DC Series computers, these manuals explain how to start, update, and change certain system parameters. The manuals also help you interpret error messages and codes. Companion manuals are *Using the AOS/VS System Management Interface* and *Using the AOS/VS II System Management Interface*.

If you have one of these computer systems, use the pertinent manual above; discard any other *Starting and Updating Preinstalled* manuals you receive.

Using the AOS/VS System Management Interface (SMI) (069–000203)

Using the AOS/VS II System Management Interface (SMI) (069–000311)

For those working with preinstalled systems and those on regular systems who want an alternative to the CLI, the SMI is an easy-to-use, menu-driven program that helps with system management functions and some file maintenance tasks.

For Programmers

AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A through ?Q (093-000542)

AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R through ?Z (093-000543)

For system programmers and application programmers who use system calls, this two-volume manual provides detailed information about system calls, including their use, syntax, accumulator input and output values, parameter packets, and error codes. *AOS/VS System Concepts* is a companion manual.

AOS/VS Debugger and File Editor User's Manual (093-000246)

For assembly language programmers, this manual describes using the AOS/VS and AOS/VS II debugger for examining program files, and the file editor FED for examining and modifying locations in any kind of disk file, including program and text files. The *AOS/VS Debug/FED* template (093-000396) accompanies this manual.

AOS/VS Link and Library File Editor (LFE) User's Manual (093-000245)

For AOS/VS and AOS/VS II programmers, this manual describes the Link utility, which builds executable program files from object modules and library files, and which can also be used to create programs to run under the AOS, MP/AOS, RDOS, RTOS, or DG/UX™ operating systems. This manual also describes the Library File Editor utility, LFE, for creating, editing, and analyzing library files; and the utilities CONVERT and MKABS, for manipulating RDOS and RTOS files.

AOS/VS Macroassembler (MASM) Reference Manual (093-000242)

For assembly language programmers, this reference manual describes the use and operation of the MASM utility, which works under AOS/VS and AOS/VS II.

AOS/VS System Concepts (093-000335)

For system programmers and application programmers who write assembly-language subroutines, this manual explains basic AOS/VS system concepts, most of which apply to AOS/VS II as well. This manual complements both volumes of the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary*.

SPEED Text Editor (AOS and AOS/VS) User's Manual (093-000197)

For programmers, this manual explains how to use SPEED, a powerful (but unforgiving) character-oriented text editor.

Other Related Documents

AOS/VS and AOS/VS II Performance Package User's Manual (093-000364)

For system managers, this manual explains how to use the AOS/VS and AOS/VS II Performance Package (Model 30718), a separate product that is useful for analyzing and perhaps improving the performance of AOS/VS and AOS/VS II systems.

Backing Up and Restoring Files With DUMP_3/LOAD_3 (093-000561)

For system managers, operators, and experienced users, this manual explains the DUMP_3/LOAD_3 product, separately available, which provides backup and enhanced restoration functions, including precise indexing of files on a backup tape set.

Configuring and Managing the High-Availability Disk-Array/MV (H.A.D.A./MV) Subsystem
(014-002160)

For system managers of the H.A.D.A./MV subsystem (a separate product), this manual explains how to configure, operate, and replace subsystem controllers, disk modules, and tape modules. This manual also explains how to replace fans, power supplies, and other subsystem hardware.

Configuring Your Network with XTS (093-00689)

For network administrators, managers, or operators responsible for designing, configuring, or maintaining a network management system, this manual describes how to manage and operate Data General's XODIAC™ Transport Service (XTS and XTS II) under AOS/VS and AOS/VS II.

Installing and Administering DG TCP/IP (093-701027)

For network managers and operators, this manual explains how to install and manage a TCP/IP network under AOS/VS.

Managing AOS/VS II ONC™ /NFS® Services (093-000667)

For network managers and operators, this manual explains how to install and manage an ONC Network File server software under AOS/VS II.

Managing AOS/VS II TCP/IP (093-000704)

For network managers and operators, this manual explains how to install and manage a TCP/IP network under AOS/VS II.

Managing and Operating the XODIAC™ Network Management System (093-000260)

For network managers and operators, this manual describes how to install and manage the Data General proprietary network software.

Managing XTS II with DG/OpenNMS (093-000698)

For network managers and operators, this manual explains how to use DG/OpenNMS to manage the XTS II transport service for large communications networks. It also identifies the XTS II components and explains how to use the NMI menus and screens to manage the XTS II subsystems and the Message Transport Agent (MTA).

Managing Your DG/PC*Integration Network with DG/ONMS (093-000624)

For network managers, this manual explains how to manage XTS II and DG/PC*Integration components with DG/OpenNMS.

Managing Your Network with DG/OpenNMS (093-000486)

For network managers, administrators, and operators, this manual describes how to use the DG/OpenNMS software. It also explains how to load the software, create the DG/OpenNMS environment, and use the Network Management Interface (NMI) to manage the network.

Managing Your XODIAC™ Network with DG/ONMS (093-000625)

For network managers, this manual explains how to manage XTS II, MTA, and the XODIAC agents (FTA, RMA, and SVTA) with DG/OpenNMS.

Programming with the Remote Procedure Call (RPC) on AOS/VS II (093–000770)

For experienced network programmers, this manual provides information necessary to write the Remote Procedure Call for the AOS/VS II UDP/IP and TCP/IP networks.

Using CLASP (Class Assignment and Scheduling Package) (093–000422)

For system managers, this manual explains how to use the AOS/VS and AOS/VS II Class Assignment and Scheduling Package (Model 31134), a separate product that is useful for tailoring process scheduling to the needs of a specific site.

Using the MV Data Center Manager (093–000769)

For system managers, this manual explains how to use the MV Data Center Manager software, a separate product that manages multiple ECLIPSE MV/Family computers from an AViON workstation.

End of Document Set

