System Call Dictionary (AOS/VS and AOS/DVS)

093-000241-03

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. 093-000241
© Data General Corporation, 1980, 1983, 1986
All Rights Reserved
Printed in the United States of America
Revision 03, September 1986
Licensed Material - Property of Data General Corporation

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 the 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.

CEO, DASHER, DATAPREP, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, INFOS, MANAP, microNOVA, NOVA, PRESENT, PROXI, SWAT and **TRENDVIEW** are U.S. registered trademarks of Data General Corporation, and **AOSMAGIC, AOS/VSMAGIC, ArrayPlus, AWE/4000, AWE/8000, AWE/10000 BusiGEN, BusiPEN, BusiTEXT, COMPUCALC, CEO Connection, CEO Drawing Board, CEO Wordview, CEOwrite, CSMAGIC, DASHER/One, DATA GENERAL/One, DESKTOP/UX, DG/GATE, DG/L, DG/XAP, DGConnect, DXA, ECLIPSE MV/10000, ECLIPSE MV/20000, ECLIPSE MV/2000, FORMA-TEXT, GDC/1000, GDC/2400, GENAP, GW/4000, GW/8000, GW/10000, microECLIPSE, MV/UX, PC Liaison, RASS, REV-UP, SPARE MAIL, UNITE,** and **XODIAC** are U.S. trademarks of Data General Corporation.

Ada is a U.S. registered stradmark of the U.S. Government (Ada Joint Program Office)

System Call Dictionary
(AOS/VS and AOS/DVS)
093-000241-03

Revision History:		Effective with:
Original Release	- November 1980	
First Revision	- March 1983	
Second Revision	- February 1986	
Third Revision	- September 1986	AOS/VS Rev. 7.50 AOS/DVS Rev. 1.00

A vertical bar in the margin of a page indicates substantive change from the first revision, the second revision, or both. An asterisk (*) in the margin of a page indicates substantive deletion from these revisions.

Preface

This manual supersedes the *Advanced Operating System/Virtual Storage (AOS/VS) Programmer's Manual, Volume 2, System Calls* (093-000241-01). It 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.

In this revision, we include new features and enhancements to existing features of the Advanced Operating System/Virtual Storage (AOS/VS) software for Revision 7.50. We also describe the features of the Advanced Operating System/Distributed Virtual Storage (AOS/DVS) operating system for its Revision 1.00.

The manual is organized as follows:

Chapter 1	introduces general concepts you'll need to write system calls and their accompanying parameter packets.
Chapter 2	contains all AOS/VS and AOS/DVS system calls in alphabetical order. It begins with a summary table of all the system calls.
Appendix A	contains the error codes for all AOS/VS and AOS/DVS system calls.
Appendix B	contains 11 sample program sets. Each set has programs that make many system calls. Ten program sets contain assembly language source programs and one contains FORTRAN 77 source programs.

Related Documentation

This manual has two companions:

- *AOS/VS System Concepts* (093-000335)
- *AOS/DVS System Concepts* (093-000416)

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), a general introduction to Data General computers.
- *Advanced Operating System/Virtual Storage (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.

The two system concepts manuals listed above contain a general description of operating system calls and how to use them. The System Call Dictionary (this book) contains detailed descriptions of each AOS/VS and AOS/DVS call. For your convenience, the system call descriptions in this manual are in alphabetical order. The error codes are in Appendix A.

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. Your system manager should be able to tell you the exact pathname of the Release Notice.

Reader Please Note:

We use the following conventions in this manual:

Symbol	Meaning
< >	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.

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.

* Contacting Data General

- If you have comments on this manual, please use the prepaid Remarks Form that appears after the Index. We want to know what you like and dislike about this manual.
- If you need additional manuals, please use the enclosed TIPS order form (USA only) or contact your local Data General sales representative.

End of Preface

Contents

Chapter 1 - Introduction to System Calls

Standard Format for System Calls 1-2
Parameter Packets 1-3
Parametric Coding 1-3
Other System Call Input 1-5
Reserved Symbols 1-6
A Complete Example — Assembly Language 1-6
  General Step 1-6
  Program Listing 1-7
  Program Construction 1-9
  Program Execution 1-9
High-level Language Interface 1-11

Chapter 2 - AOS/VS and AOS/DVS System Calls

?ALLOCATE	Allocates disk blocks	2-16
?ASSIGN	Assigns a character device to a process	2-18
?ASSOC/?DISSOC	Associates/Dissociates two processes for passing obituary messages	2-19
?AWIRE	Changes the wiring characteristics of the Agent	2-24
?BLKIO	Performs (reads/writes) block I/O	2-25
?BLKPR	Blocks a process	2-32
?BNAME	Determines whether process name/queue name is on local or remote host	2-34
?BRKFL	Terminates a process and creates a break file	2-35
?CDAY	Converts a scalar date value	2-38
?CGNAM	Gets a complete pathname from a channel number	2-39
?CHAIN	Passes control from a Ring 7 caller to a new program	2-40
?CKVOL	Checks volume identifier of a labeled magnetic tape	2-42
?CLASS	Gets or sets class IDs	2-43
?CLOSE	Closes an open channel	2-45
?CLRDV	Clears a device	2-47
?CLSCHED	Enables, disables, or examines class scheduling	2-49
?CLSTAT	Returns class scheduling statistics	2-53
?CMATRIX	Gets or sets the class matrix	2-59
?CON	Becomes a customer of a specified server	2-64
?CPMAX	Sets maximum size for a control point directory (CPD)	2-66

?CREATE	Creates a file or directory	2-68
?CRUDA	Creates a user data area (UDA)	2-78
?CTERM	Terminates a customer process	2-80
?CTOD	Converts a scalar time value	2-82
?CTYPE	Changes a process type	2-83
?DACL	Sets, clears, or examines a default access control list	2-85
?DADID	Gets the PID of a process's father	2-87
?DCON	Breaks a connection (disconnects) in Ring 7	2-88
?DDIS	Disables access to all devices	2-90
?DEASSIGN	Cancels a character device	2-91
?DEBL	Enables access to all devices	2-92
?DEBUG	Calls the Debugger utility	2-93
?DELAY	Suspends a 16-bit task for a specified interval (16-Bit Processes Only)	2-94
?DELETE	Deletes a file entry	2-95
?DEREGISTER	Deletes an entry from the Public Name Registry	2-97
?DFRSCH	Disables task rescheduling and indicates prior state of rescheduling	2-99
?DIR	Changes the working directory	2-100
?DISSOC	Disassociates two processes	2-101
?DQTSK	Removes from the queue one or more previously queued tasks	2-102
?DRCON	Breaks a connection (disconnects)	2-104
?DRSCH	Disables scheduling	2-105
?ENBRK	Enables a break file	2-106
?ENQUE	Queues a file entry	2-110
?ERMSG	Reads the error message file	2-112
?ERSCH	Enables multitask scheduling for the calling process	2-115
?ESFF	Flushes shared file memory pages to disk	2-116
?EXEC	Requests a service from EXEC	2-117
?EXPO	Sets, clears, or examines execute-protection status	2-148
?FDAY	Converts date to a scalar value	2-150
?FEDFUNC	Interfaces to File Editor (FED) utility	2-151
?FEOV	Forces end-of-volume on labeled magnetic tape	2-156
?FLOCK	Locks a file element	2-157
?FLUSH	Flushes the contents of a shared page to disk	2-159
?FSTAT	Gets file status information	2-160
?FTOD	Converts time of day to a scalar value	2-168
?FUNLOCK	Unlocks a file element	2-169
?GACL	Gets a file entry's access control list (ACL)	2-171
?GBIAS	Gets the current bias factor values	2-173
?GCHR	Reads device characteristics of a character device	2-174
?GCLOSE	Closes a file previously opened for block I/O	2-180
?GCPN	Gets the console port number	2-182
?GCRB	Gets the base of the current resource (16-Bit Processes Only)	2-184
?GDAY	Gets the current date	2-185
?GDLM	Gets a delimiter table	2-186
?GECHR	Get extended characteristics	2-188
?GHRZ	Gets the frequency of the system clock	2-190
?GLINK	Gets the contents of a link entry	2-191
?GLIST	Gets the contents of a search list	2-192
?GMEM	Returns the number of undedicated memory pages	2-193

?GNAME	Gets a complete pathname	2-194
?GNFN	Lists a particular directory's entries	2-195
?GOPEN	Opens a file for block I/O	2-198
?GPID	Returns all active PIDs based on a host ID	2-205
?GPORT	Returns the PID associated with a global port number	2-208
?GPOS	Gets the current file-pointer position	2-209
?GPRNM	Gets a program's pathname	2-212
?GRAPHICS	Manipulates pixel maps	2-213
?GRNAME	Returns complete pathname of generic file	2-232
?GSHPT	Lists the current shared partition size	2-233
?GSID	Gets the system identifier	2-234
?GTACP	Gets access control privileges	2-235
?GTMES	Gets a CLI message	2-237
?GTNAM	Returns symbol closest in value to specified input value	2-245
?GTOD	Gets the time of day	2-247
?GTRUNCATE	Truncates a disk file	2-248
?GTSVL	Gets the value of a user symbol	2-250
?GUNM	Gets the username of a process	2-252
?GVPID	Gets the virtual PID of a process	2-254
?HNAME	Gets a host name or host identifier	2-255
?IDEF	Defines a user device	2-257
?IDGOTO	Redirects a task's execution path	2-264
?IDKIL	Kills a task specified by its TID	2-265
?IDPRI	Changes the priority of a task specified by its TID	2-266
IDRDY	Readies a task specified by its TID	2-267
?IDSTAT	Returns task status word (16-Bit Processes Only)	2-268
?IDSUS	Suspends a task specified by its TID	2-269
?IESS	Initializes an extended state save (ESS) area (16-Bit Processes Only)	2-270
?IFPU	Initializes the floating-point unit	2-271
?IHIST	Starts a histogram for a 16-bit process (16-Bit Processes Only)	2-272
?ILKUP	Returns a global port number	2-275
?IMERGE	Modifies a ring field within a global port number	2-276
?IMSG	Receives an interrupt service message	2-277
?INIT	Initializes a logical disk	2-278
?INTWT	Defines a console interrupt task	2-280
?IQTSK	Creates a queued task manager	2-281
?IREC	Receives an IPC message	2-282
?IRMV	Removes a user device	2-292
?ISEND	Sends an IPC message	2-293
?ISPLIT	Finds the owner of a port (including its ring number)	2-296
?IS.R	Sends and then receives an IPC message	2-297
?ITIME	Returns the OS-format internal time	2-301
?IXIT	Exits from an interrupt service routine	2-302
?IXMT	Transmits a message from an interrupt service routine	2-303
?JPINIT	Initializes a job processor	2-304
?JPMOV	Moves a job processor to a new logical processor	2-307
?JPREL	Releases a job processor	2-309
?JPSTAT	Gets the status of a job processor	2-311

?KCALL	Keeps the calling resource and acquires a new resource (16-Bit Processes Only)	2-315
?KHIST	Kills a histogram	2-316
?KILAD	Defines a kill-processing routine	2-317
?KILL	Kills the calling task	2-318
?KINTR	Simulates keyboard interrupt sequences	2-319
?KIOFF	Disables control-character console interrupts	2-320
?KION	Re-enables control-character console interrupts	2-321
?KWAIT	Waits for a console interrupt	2-322
?LABEL	Creates a label for a magnetic tape	2-323
?LEFD	Disables LEF mode	2-327
?LEFE	Enables LEF mode	2-328
?LEFS	Returns the current LEF mode status	2-329
?LMAP	Maps a lower ring	2-330
?LOCALITY	Changes user locality	2-332
?LOGCALLS	Logs system calls	2-335
?LOGEV	Enters an event in the system log file	2-338
?LPCLASS	Gets/sets logical processor class assignments	2-340
?LPCREA	Creates a logical processor	2-343
?LPDELE	Deletes a logical processor	2-345
?LPSTAT	Gets the status of a logical processor	2-347
?MAPDV	Maps a device into logical address space	2-352
?MBFC	Moves bytes from a customer's buffer	2-356
?MBTC	Moves bytes to a customer's buffer	2-358
?MDUMP	Dumps the memory image from a user-specified ring to a file	2-360
?MEM	Lists the current unshared memory parameters	2-362
?MEMI	Changes the number of unshared pages in the logical address space	2-363
?MIRROR	Mirrors and synchronizes LDU images	2-365
?MPHIST	Starts a histogram on a uni- or multiprocessor system (32-Bit Processes Only)	2-372
?MYTID	Gets the priority and TID of the calling task	2-377
?ODIS	Disables console interrupts	2-378
?OEBL	Enables console interrupts	2-379
?OPEN	Opens a file	2-380
?OPER	Creates and maintains an operator interface	2-399
?OPEX	Communicates between the current process and an operator process	2-413
?OVEX	Releases an overlay and returns (16-Bit Processes Only)	2-496
?OVKIL	Exits from an overlay and kills the calling task (16-Bit Processes Only)	2-497
?OVLOD	Loads and goes to an overlay (16-Bit Processes Only)	2-498
?OVREL	Releases an overlay area (16-Bit Processes Only)	2-501
?PCLASS	Gets a process's class and locality	2-502
?PCNX	Passes a connection from one server to another in Ring 7	2-505
?PGLIST	Gets a process group name search list	2-506
?PGNAME	Returns a process group's identifier	2-507
?PIDS	Gets information about PIDs	2-510
?PMTPF	Permits access to a protected file	2-512
?PNAME	Gets a full process name	2-514

?PRCNX	Passes a connection from one server to another	2-516
?PRDB/?PWRB	Performs physical block I/O	2-518
?PRI	Changes the priority of the calling task	2-523
?PRIPR	Changes the priority of a process	2-524
?PRKIL	Kills all tasks of a specified priority	2-526
?PROC	Creates a process	2-527
?PROFILE	Performs a profile request	2-543
?PRRDY	Readies all tasks of a specified priority	2-551
?PRSUS	Suspends all tasks of a specified priority	2-552
?PSLIST	Sets a process group name search list	2-553
?PSTAT	Returns status information on a process	2-555
?PTRDEVICE	Controls input from a pointer device	2-564
?PWDCRYP	Performs a password data encryption request	2-580
?PWRB	Performs physical block I/O	2-583
?RCALL	Releases one resource and acquires a new one (16-Bit Processes Only)	2-584
?RCHAIN	Chains to a new procedure (16-Bit Processes Only.)	2-585
?RDB/?WRB	Performs (reads/writes) block I/O	2-586
?RDUDA/?WRUDA	Reads/writes a user data area (UDA)	2-593
?READ/?WRITE	Performs (reads/writes) record I/O	2-595
?REC	Receives an intertask message	2-612
?RECNW	Receives an intertask message without waiting	2-613
?RECREATE	Recreates a file	2-614
?REGISTER	Creates an entry in the Public Name Registry	2-615
?RELEASE	Releases an initialized logical disk	2-618
?RENAME	Renames a file	2-619
?RESCHED	Reschedules current time slice (32-Bit Processes Only)	2-621
?RESIGN	Resigns as a server	2-622
?RETURN	Terminates the calling process and passes the termination message to the father	2-623
?RINGLD	Loads a program file into a specified ring	2-625
?RNAME	Determines whether a pathname contains a reference to a remote host	2-627
?RNGPR	Returns the .PR filename for a ring	2-628
?RNGST	Stops ?RINGLD from loading lower rings	2-630
?RPAGE	Releases a shared page	2-631
?RUNTM	Gets runtime statistics on a process	2-633
?SACL	Sets a new access control list (ACL)	2-636
?SATR	Sets or removes the permanent attribute for a file or directory	2-639
?SBIAS	Sets the bias factors	2-641
?SCHR	Sets a character device's characteristics	2-643
?SCLOSE	Closes a file previously opened for shared access	2-647
?SDAY	Sets the system calendar	2-649
?SDBL	Disables a BSC line	2-650
?SDLM	Sets a delimiter table	2-651
?SDPOL	Defines a polling list or a poll-address/select-address pair	2-653
?SDRT/?SERT	Disables/re-enables a relative console	2-656
?SEBL	Enables a BSC line	2-658
?SECHR	Sets extended characteristics of a device	2-667
?SEND	Sends a message to a console	2-669
?SERMSG	Returns text for associated error code (16-Bit Processes Only)	2-671

?SERT	Re-enables a relative console	2-673
?SERVE	Becomes a server	2-674
?SGES	Gets BSC error statistics	2-675
?SIGNL	Signals another task	2-677
?SIGWT	Signals another task and then waits for a signal	2-679
?SINFO	Gets selected information about the current operating system	2-681
?SLIST	Sets the search list for the calling process	2-684
?SONS	Gets a list of son processes for a target PID	2-685
?SOPEN	Opens a file for shared access	2-688
?SOPPF	Opens a protected shared file	2-690
?SPAGE	Performs a shared-page read	2-693
?SPOS	Sets the position of the file pointer	2-696
?SRCV	Receives data or a control sequence over a BSC line	2-698
?SSHPT	Establishes a new shared partition	2-708
?SSID	Sets the system identifier	2-709
?SSND	Sends data or a control sequence over a BSC line	2-710
?STMAP	Sets the data channel map	2-720
?STOD	Sets the system clock	2-723
?STOM	Sets the time-out value for a device	2-724
?SUPROC	Enters, leaves, or examines Superprocess mode	2-726
?SUS	Suspends the calling task	2-728
?SUSER	Enters, leaves, or examines Superuser mode	2-729
?SYLOG	Manipulates the system log file	2-731
?SYSPRV	Enters, leaves, or examines a privilege state	2-733
?TASK	Initiates one or more tasks	2-736
?TERM	Terminates a process	2-745
?TIDSTAT	Returns status of target task (32-Bit Processes Only)	2-747
?TLOCK	Protects a task from being redirected	2-748
?TMSG	Defines the termination message format	2-750
?TPID	Translates a PID	2-759
?TPORT	Translates a local port number to its global equivalent	2-760
?TRCON	Reads a task message from the process console	2-761
?TRUNCATE	Truncates a file at the current position	2-763
?TUNLOCK	Allows a task to be redirected	2-764
?UBLPR	Unblocks a process	2-766
?UIDSTAT	Returns the status of a task and an unambiguous identifier	2-768
?UNWIND	Unwinds the stack and restores the previous environment (16-Bit Processes Only)	2-770
?UNWIRE	Unwires pages previously wired	2-771
?UPDATE	Flushes file descriptor information	2-772
?VALAD	Validates a logical address	2-773
?VALIDATE	Validates an area for Read or Write access	2-774
?VCUST	Verifies a customer in Ring	2-778
?VRCUST	Verifies a customer in a specified ring	2-779
?WALKBACK	Returns information about previous frames in the stack (16-Bit Processes Only)	2-780
?WDELAY	Suspends a task for a specified time (32-Bit Processes Only)	2-782
?WHIST	Starts a histogram (32-Bit Processes Only)	2-783
?WINDOW	Manipulates windows	2-786
?WIRE	Wires pages to the working set	2-820
?WRB	Writes block I/O	2-822

?WRITE Writes record I/O 2-823
?WRUDA Writes a user data area (UDA) 2-824
?WTSIG Waits for a signal from another task or process 2-825
?XINIT Initializes a logical disk (extended) 2-827
?XMT Transmits an intertask message 2-837
?XMTW Transmits an intertask message and waits for it to be received .. 2-838
?XPSTAT Returns extended status information on a process 2-839

Appendix A - Error Codes

Appendix B - Sample Programs

Program Set 1 — HEAR.SR, SPEAK.SR, SON.SR B-3
Program Set 2 — RUNTIME.SR B-12
Program Set 3 — RINGLOAD.SR, INRING.SR, and GATE.ARRAY.SR B-15
Program Set 4 — FILCREATE.SR B-20
Program Set 5 — WRITE.SR B-23
Program Set 6 — DLIST.SR B-27
Program Set 7 — NEWTASK.SR B-30
Program Set 8 — BOOMER.SR B-33
Program Set 9 — TIMEOUT.SR B-38
Program Set 10 — DIRCREATE.F77 and CHECK.F77 ..... B-40
General Steps B-40
Program Listings B-41
Program Construction B-47
Program Execution B-47
Program Set 11 — CREATE_WINDOW.SR. B-49

Figures

Figure	Caption	
1-1	Parametric Coding Example	1-3
1-2	Listing File of Program DIRCREATE.SR	1-7
2-1	Structure of ?ASSOC Packet	2-21
2-2	Structure of ?DISSOC Packet	2-22
2-3	Structure of ?BLKIO Packet	2-26
2-4	Examples of Read Next Allocated Element Option	2-29
2-5	Structure of Physical I/O Controller Status Block	2-31
2-6	Structure of ?CLASS Packet	2-44
2-7	Structure of ?CLOSE Packet	2-46
2-8	Structure of ?CLSCHED Packet	2-51
2-9	Structure of ?CLSTAT Main Packet	2-55
2-10	Structure of ?CLSTAT Subpacket	2-57
2-11	Structure of ?CMATRIX Main Packet	2-61
2-12	Structure of ?CMATRIX Subpacket	2-62
2-13	Addresses of ?CMATRIX Main Packet and Its Subpacket Offsets	2-63
2-14	Structure of ?CPMAX packet	2-67
2-15	Structure of ?CREATE IPC Packet	2-70
2-16	Structure of ?CREATE Time Block	2-71
2-17	Structure of ?CREATE Directory Packet	2-72
2-18	Structure of ?CREATE Packet for Other File Types	2-74
2-19	Structure of ?CRUDA Packet	2-79
2-20	Structure of ?DELETE Packet	2-96
2-21	Extended Task Definition Packet	2-103
2-22	Structure of ?ENBRK Packet	2-108
2-23	Error Code Structure in ERMES File	2-113
2-24	Structure of ?EXEC Packet for Unlabeled Mount Function ?XFMUN	2-119
2-25	Structure of ?EXEC Extended Packet for Unlabeled Mount Function ?XFXUN	2-119

Licensed Material - Property of Data General Corporation

Figure	Caption	
2-26	Structure of ?EXEC Packet for Labeled Mount Function ?XFMLT	2-120
2-27	Structure of ?EXEC Extended Packet for Labeled Mount Function ?XFXML	2-120
2-28	Structure of ?EXEC Packet for Dismounting a Tape, ?XFDUN	2-121
2-29	Structure of ?EXEC Packet for Queue Requests	2-123
2-30	Structure of AOS/VS ?EXEC Packet for Holding, Unholding, or Canceling Queue Requests	2-131
2-31	Structure of AOS/DVS ?EXEC Packet for Holding, Unholding, or Canceling Queue Requests	2-133
2-32	Structure of ?EXEC Packet for Status Information	2-135
2-33	Structure of ?EXEC Packet for Extended Status Information	2-136
2-34	Structure of ?EXEC Packet for a MOUNT Queue Request	2-138
2-35	Structure of ?EXEC Packet for Dismounting a Unit (Extended Request)	2-139
2-36	Structure of ?EXEC Packet for Changing Queuing Parameters	2-140
2-37	Structure of ?EXEC Packet for Obtaining Queue Names	2-144
2-38	Structure of ?EXEC Packet for Obtaining QDISPLAY Information	2-146
2-39	Structure of ?FEDFUNC Packet to Change Radix	2-152
2-40	Structure of ?FEDFUNC Packet to Open Symbol Table File	2-153
2-41	Structure of ?FEDFUNC Packet to Evaluate a FED String	2-153
2-42	Structure of ?FEDFUNC Packet to Disassemble an Instruction	2-154
2-43	Structure of ?FEDFUNC Packet to Insert a Temporary Symbol	2-154
2-44	Structure of ?FEDFUNC Packet to Delete a Temporary Symbol	2-155
2-45	Structure of ?FLOCK Packet	2-158
2-46	Structure of ?FSTAT Packet for Unit Files	2-162
2-47	Structure of ?FSTAT Packet for IPC Files	2-163
2-48	Structure of ?FSTAT Packet for Directory Files	2-164
2-49	Structure of ?FSTAT Packet for Other File Types	2-165
2-50	?SSTS Structure	2-166
2-51	Structure of ?FUNLOCK Packet	2-170
2-52	Structure of ?GACL Packet	2-172
2-53	Structure of ?GNFN Packet	2-196
2-54	Structure of ?GOPEN Packet for IPC Files	2-199
2-55	Structure of Standard ?GOPEN Packet	2-200
2-56	Structure of ?GOPEN Packet Extension	2-202
2-57	Structure of ?GPID Packet	2-206
2-58	Standard I/O Packet for ?READ, ?WRITE, ?OPEN and ?CLOSE	2-210
2-59	Structure of the ?GRAPHICS Main Packet	2-215
2-60	Structure of the ?GRAPH_OPEN_WINDOW_PIXELMAP Subpacket	2-217
2-61	Structure of the ?GRAPHICS_CREATE_MEMORY_PIXELMAP	2-219
2-62	Structure of the ?GRAPH_PIXELMAP_STATUS Subpacket	2-222
2-63	Structure of the ?GRAPH_SET_CLIP_RECTANGLE Subpacket	2-224
2-64	Structure of the ?GRAPH_MAP_PIXELMAP Subpacket	2-227
2-65	Structure of the ?GRAPH_WRITE_PALETTE Subpacket	2-230

Figure	Caption	
2-66	Structure of the ?GRAPH_READ_PALETTE Subpacket	2-231
2-67	Structure of ?GTMES Packet	2-238
2-68	Structure of ?GTRUNCATE Packet	2-249
2-69	Structure of Device Control Table (DCT) for 32-Bit Processes	2-259
2-70	Structure of Device Control Table (DCT) for 16-Bit Processes	2-260
2-71	Structure of Map Definition Table	2-261
2-72	Structure of ?IHIST Packet	2-273
2-73	Structure of ?IREC Header	2-283
2-74	Structure of Offset ?IUFL	2-285
2-75	Structure of ?ISEND Header	2-294
2-76	Structure of ?IS.R Header	2-298
2-77	Structure of ?JPINIT Packet	2-305
2-78	Structure of ?JPMOV Packet	2-308
2-79	Structure of ?JPREL Packet	2-310
2-80	Structure of ?JPSTAT Main Packet	2-312
2-81	Structure of ?JPSTAT General Information Subpacket	2-313
2-82	Structure of ?JPSTAT Specific Information Subpacket	2-314
2-83	Structure of ?LABEL Packet	2-324
2-84	Structure of ?LOCALITY Packet	2-333
2-85	?LOGEV Event Logging Format	2-339
2-86	Structure of ?LPCLASS Packet	2-341
2-87	Structure of ?LPCREA Packet	2-344
2-88	Structure of ?LPDELE Packet	2-346
2-89	Structure of ?LPSTAT Main Packet	2-349
2-90	Structure of ?LPSTAT General Information Subpacket	2-350
2-91	Structure of ?LPSTAT Specific Information Subpacket	2-351
2-92	Structure of ?MAPDV Packet	2-354
2-93	Structure of ?MBFC Packet	2-357
2-94	Structure of ?MBTC Packet	2-359
2-95	Structure of ?MIRROR Packet	2-366
2-96	Structure of ?MIRROR Subpacket	2-368
2-97	Structure of 16-Bit ?MIRROR Packet	2-369
2-98	Structure of 16-Bit ?MIRROR Subpacket	2-371
2-99	Structure of ?MPHIST Packet	2-374
2-100	Structure of ?OPEN Packet	2-382
2-101	Sample Delimiter Table	2-389
2-102	Structure of ?OPEN Extension Packet for Pipes	2-390
2-103	Structure of Labeled Magnetic Tape Packet Extension	2-393
2-104	Structure of ?OPER Main Packet	2-400
2-105	Structure of ?OPON Subpacket	2-402
2-106	Structure of ?OPOFF Subpacket	2-404
2-107	Structure of ?OPSEND Subpacket	2-406
2-108	Structure of ?OPRCV Subpacket	2-408
2-109	Structure of ?OPRESP Subpacket	2-410
2-110	Structure of ?OPINFO Subpacket	2-411

Figure	Caption	
2-111	Structure of ?OPEX Main Packet	2-414
2-112	Structure of Access Command Subpacket	2-418
2-113	Structure of Align Command Subpacket	2-419
2-114	Structure of Batch_List Command Subpacket	2-421
2-115	Structure of Batch_Output Command Subpacket	2-422
2-116	Structure of Binary Command Subpacket	2-423
2-117	Structure of Brief Command Subpacket	2-424
2-118	Structure of Cancel Command Subpacket	2-425
2-119	Structure of Consolestatus Command Subpacket	2-427
2-120	Structure of Continue Command Subpacket	2-429
2-121	Structure of CPL Command Subpacket	2-430
2-122	Structure of Create Command Subpacket	2-431
2-123	Structure of Defaultforms Command Subpacket	2-432
2-124	Structure of Disable Command Subpacket	2-434
2-125	Structure of Dismounted Command Subpacket	2-435
2-126	Structure of Elongate Command Subpacket	2-436
2-127	Structure of Enable Command Subpacket	2-437
2-128	Structure of Even Command Subpacket	2-439
2-129	Structure of Flush Command Subpacket	2-440
2-130	Structure of Forms Command Subpacket	2-442
2-131	Structure of Halt Command Subpacket	2-443
2-132	Structure of Headers Command Subpacket	2-444
2-133	Structure of Hold Command Subpacket	2-445
2-134	Structure of Limit Command Subpacket	2-446
2-135	Structure of Logging Command Subpacket	2-447
2-136	Structure of LPP Command Subpacket	2-449
2-137	Structure of Mounted Command Subpacket	2-453
2-138	Structure of Mountstatus Command Subpacket	2-454
2-139	Structure of Operator Command Subpacket	2-458
2-140	Structure of Pause Command Subpacket	2-459
2-141	Structure of Premount Command Subpacket	2-460
2-142	Structure of Priority Command Subpacket	2-461
2-143	Structure of Prompts Command Subpacket	2-462
2-144	Structure of Qpriority Command Subpacket	2-464
2-145	Structure of Refused Command Subpacket	2-466
2-146	Structure of Release Command Subpacket	2-467
2-147	Structure of Restart Command Subpacket	2-468
2-148	Structure of Silence Command Subpacket	2-469
2-149	Structure of Spoolstatus Command Subpacket	2-470
2-150	Structure of Stack Command Subpacket	2-473

Figure	Caption	
2-151	Structure of Start Command Subpacket	2-474
2-152	Structure of Status Command Subpacket	2-477
2-153	Structure of Stop Command Subpacket	2-481
2-154	Structure of Trailers Command Subpacket	2-483
2-155	Structure of Unhold Command Subpacket	2-484
2-156	Structure of Unitstatus Command Subpacket	2-485
2-157	Structure of Unlimit Command Subpacket	2-487
2-158	Structure of Unsilence Command Subpacket	2-488
2-159	Structure of Verbose Command Subpacket	2-489
2-160	Structure of Xbias Command Subpacket	2-490
2-161	Structure of Autokey Command Subpacket	2-491
2-162	Structure of Register Command Subpacket	2-492
2-163	Structure of Registerstatus Command Subpacket	2-494
2-164	Structure of ?PCLASS Packet	2-503
2-165	Structure of ?PGNAME Packet	2-508
2-166	Structure of ?PIDS Packet	2-511
2-167	Structure of ?PMTPF Packet	2-513
2-168	Structure of ?PRDB/?PWRB Packet	2-519
2-169	Structure of ?PROC Packet	2-530
2-170	Structure of ?PROC Extension Packet	2-539
2-171	Structure of ?PROFILE Parameter Packet	2-544
2-172	Structure of ?PROFILE Field Descriptor Packet	2-546
2-173	Structure of ?PSTAT Packet	2-557
2-174	Structure of ?PSTAT Memory Descriptor	2-558
2-175	Structure of the ?PTRDEVICE Main Packet	2-566
2-176	Structure of the ?PTRDEV_SET_EVENTS Subpacket	2-568
2-177	Structure of the ?PTRDEV_SET_DELTA Subpacket	2-570
2-178	Structure of the ?PTRDEV_LAST_EVENT Subpacket	2-572
2-179	Subpacket for ?PTRDEV_SET_POINTER Subpacket	2-573
2-180	Structure of the ?PTRDEV_GET_PTR_STATUS Subpacket	2-575
2-181	Structure of the ?PTRDEV_GENERATE_EVENT Subpacket	2-578
2-182	Structure of ?PWDCRYP Packet	2-581
2-183	Structure of ?RDB/?WRB Packet	2-587
2-184	Structure of ?RDUDA/?WRUDA Packet	2-594
2-185	Structure of ?READ/?WRITE Packet	2-596
2-186	Structure of Screen-Management Packet Extension	2-603
2-187	Structure of Selected Field Translation Packet Extension	2-608
2-188	Selected Field Translation Packet Sample Listing	2-610
2-189	Structure of ?REGISTER Packet	2-616
2-190	Structure of ?RENAME Packet	2-620

Figure	Caption	
2-191	Structure of the ?RNGPR Packet	2-629
2-192	Structure of ?RUNTM Packet	2-634
2-193	Structure of ?SACL Packet	2-638
2-194	Structure of ?SATR Packet	2-640
2-195	Polling List Defined by a Control Station	2-655
2-196	Poll and Select Addresses Defined by a Tributary	2-655
2-197	Structure of ?SEBL Packet	2-659
2-198	Structure of the ?SERMSG Packet	2-672
2-199	Structure of ?SGES Packet	2-675
2-200	Structure of ?SINFO Packet	2-682
2-201	Structure of ?SONS Packet	2-686
2-202	Structure of ?SOPPF Packet	2-691
2-203	Structure of ?SPAGE Packet	2-694
2-204	Structure of ?SRCV Packet	2-699
2-205	ITB Receive Buffer Format	2-707
2-206	Structure of ?SSND Packet	2-712
2-207	Structure of ?SYSPRV Packet	2-734
2-208	Structure of Standard Task Definition Packet	2-737
2-209	Stack Parameters for Initiating One or More Tasks	2-740
2-210	Extended Task Definition Packet	2-741
2-211	Structure of Termination Message a 32-bit Process Receives	2-751
2-212	Structure of Termination Message a 16-bit Process Receives	2-754
2-213	Structure of ?UIDSTAT Packet	2-769
2-214	Structure of ?WHIST Packet	2-784
2-215	Structure of the ?WINDOW Main Packet	2-790
2-216	Structure of the ?WIN_CREATE_WINDOW Subpacket	2-792
2-217	Structure of the ?WIN_DEFINE_PORTS Subpacket	2-797
2-218	Structure of the ?WIN_SET_USER_INTERFACE Subpacket	2-803
2-219	Structure of the ?WIN_GET_USER_INTERFACE Subpacket	2-804
2-220	Structure of the ?WIN_GET_TITLE Subpacket	2-806
2-221	Structure of the ?WIN_GTITLE Subpacket	2-807
2-222	Structure of the ?WIN_WINDOW_STATUS Subpacket	2-809
2-223	Structure of the ?WIN_DEVICE_STATUS Subpacket	2-813
2-224	Structure of the ?WIN_RETURN_GROUP_WINDOWS Subpacket	2-815
2-225	Structure of the ?WIN_RETURN_DEVICE_WINDOWS Subpacket	2-816
2-226	Structure of the ?WIN_WRITE_PALETTE Subpacket	2-818
2-227	Structure of the ?WIN_READ_PALETTE Subpacket	2-819
2-228	Structure of ?XINIT Packet	2-829
2-229	Structure of ?XINIT Subpacket	2-831
2-230	Structure of 16-Bit ?XINIT Packet	2-833
2-231	Structure of 16-Bit ?XINIT Subpacket	2-836
2-232	Structure of ?XPSTAT Packet	2-841
2-233	Structure of ?XPSTAT Memory Descriptor	2-843

Figure	Caption	
B-1	Listing of Program HEAR.SR	B-3
B-2	Listing of Program SPEAK.SR	B-7
B-3	Listing of Program SON.SR	B-9
B-4	Listing of Program RUNTIME.SR	B-12
B-5	Listing of Program RINGLOAD.SR	B-15
B-6	Listing of Program INRING.SR	B-17
B-7	Listing of Program GATE.ARRAY.SR	B-18
B-8	Listing of Program FILCREATE.SR	B-20
B-9	Listing of Program WRITE.SR	B-23
B-10	Listing of Program DLIST.SR	B-27
B-11	Listing of Program NEWTASK.SR	B-30
B-12	Listing of Program BOOMER.SR	B-33
B-13	Listing of Program TIMEOUT.SR	B-38
B-14	Listing of Program DIRCREATE.F77	B-42
B-15	File DIRCREATE_SYMBOLS	B-43
B-16	File DIRCREATE_SYMBOLS.F77.IN	B-44
B-17	Subroutine CHECK.F77	B-45
B-18	File CHECK_SYMBOLS	B-46
B-19	File CHECK_SYMBOLS.F77.IN	B-46
B-20	Listing of Program CREATE_WINDOW.SR	B-49

Licensed Material - Property of Data General Corporation

Tables

Table	Caption	
2-1	Summary of AOS/VS and AOS/DVS System Calls	2-4
2-2	Contents of ?ASSOC Packet	2-21
2-3	Contents of ?DISSOC Packet	2-22
2-4	Contents of the ?BLKIO Packet	2-27
2-5	Contents of ?CLASS Packet	2-44
2-6	Contents of ?CLSCHED Packet	2-51
2-7	Contents of ?CLSTAT Main Packet	2-56
2-8	Contents of ?CLSTAT Subpacket	2-58
2-9	Contents of ?CMATRIX Main Packet	2-61
2-10	Contents of ?CMATRIX Subpacket	2-62
2-11	Valid ?CREATE File Types	2-69
2-12	Contents of ?CREATE IPC Packet	2-71
2-13	Contents of ?CREATE Directory Packet	2-73
2-14	Contents of ?CREATE Packet for Other File Types	2-75
2-15	Contents of ?ENBRK Packet	2-108
2-16	Switches for ?ENQUE Specifications String	2-111
2-17	Flags for EXEC Functions ?XFXUN and ?XFXML	2-121
2-18	Contents of ?EXEC Packet for Queue Requests	2-124
2-19	Contents of AOS/VS ?EXEC Packet for Holding, Unholding, or Canceling Queue Requests	2-132
2-20	Contents of AOS/DVS ?EXEC Packet for Holding, Unholding, or Canceling Queue Requests	2-134
2-21	Contents of ?EXEC Packet for Status Information	2-135
2-22	Contents of ?EXEC Packet for Extended Status Information	2-137
2-23	Contents of Selected Offsets in the ?EXEC Packet for the ?XFMOD Function	2-142
2-24	Contents of ?FEDFUNC Packet to Evaluate a FED String	2-155
2-25	Contents of ?FLOCK Packet	2-158

Table	Caption	
2-26	Flags Returned in Offset ?SSTS	2-166
2-27	Contents of ?FUNLOCK Packet	2-170
2-28	Character Device Characteristics Words	2-175
2-29	Commonly Used Device Characteristics	2-177
2-30	Contents of ?GNFN Packet	2-196
2-31	Contents of ?GOPEN Packet for IPC Files	2-199
2-32	Contents of Standard ?GOPEN Packet	2-200
2-33	Option Flags for Offset ?ODF1	2-203
2-34	Valid Format Options (DPM disks)	2-203
2-35	Contents of ?GPID Packet	2-207
2-36	Positioning the File Pointer	2-211
2-37	?GRAPHICS Function Codes	2-214
2-38	Contents of the ?GRAPHICS Main Packet	2-216
2-39	Contents of the ?GRAPH_OPEN_WINDOW_PIXELMAP Subpacket	2-218
2-40	Contents of the ?GRAPH_CREATE_MEMORY_PIXELMAP Subpacket	2-220
2-41	Contents of the ?GRAPH_PIXELMAP_STATUS Subpacket	2-223
2-42	Contents of the ?GRAPH_SET_CLIP.RECTANGLE Subpacket	2-225
2-43	Contents of the ?GRAPH_MAP_PIXELMAP Subpacket	2-228
2-44	Contents of ?GTMES Packet	2-239
2-45	Input Parameters for Offset ?GREQ (Request Types)	2-240
2-46	Output from ?GTMES Requests	2-242
2-47	Contents of ?GTRUNCATE Packet	2-249
2-48	Contents of Map Definition Table Entry	2-262
2-49	Structure of ?IHIST Array	2-274
2-50	Contents of ?INIT Packet	2-279
2-51	Contents of ?IREC Header	2-284
2-52	Termination Codes for 16-Bit Processes	2-285
2-53	Process Termination Codes in Offset ?IUFL for ?IREC and ?ISEND	2-286
2-54	?TRAP Termination Messages for 16-Bit Processes	2-288
2-55	?TEXT Code Termination Messages Sent on 32-Bit Process User Trap	2-290
2-56	Contents of ?ISEND Header	2-294
2-57	Contents of ?IS.R Header	2-299
2-58	Contents of ?JPINIT Packet	2-306
2-59	Contents of ?JPMOV Packet	2-308
2-60	Contents of ?JPREL Packet	2-310
2-61	Contents of ?JPSTAT Main Packet	2-312
2-62	Contents of ?JPSTAT General Information Subpacket	2-313
2-63	Contents of ?JPSTAT Specific Information Subpacket	2-314
2-64	Contents of ?LABEL Packet	2-325
2-65	Contents of ?LOCALITY Packet	2-334

Table	Caption	
2-66	Contents of ?LPCLASS Packet	2-342
2-67	Contents of ?LPCREA Packet	2-344
2-68	Contents of ?LPDELE Packet	2-346
2-69	Contents of ?LPSTAT Main Packet	2-349
2-70	Contents of ?LPSTAT General Information Subpacket	2-350
2-71	Contents of ?LPSTAT Specific Information Subpacket	2-351
2-72	Contents of ?MAPDV Packet	2-355
2-73	Contents of ?MIRROR Packet	2-367
2-74	Contents of 16-Bit ?MIRROR Packet	2-370
2-75	Contents of ?MPHIST Packet	2-375
2-76	Structure and Contents of ?MPHIST Histogram Array	2-376
2-77	Contents of ?OPEN Packet	2-383
2-78	File Creation Options for Offset ?ISTI	2-387
2-79	Common File Types You Can Create with ?OPEN	2-388
2-80	Contents of ?OPEN Extension Packet for Pipes	2-391
2-81	Contents of Labeled Magnetic Tape Packet Extension	2-394
2-82	Contents of ?OPER Main Packet	2-401
2-83	Contents of ?OPON Subpacket	2-403
2-84	Contents of ?OPOFF Subpacket	2-405
2-85	Contents of ?OPSEND Subpacket	2-407
2-86	Contents of ?OPRCV Subpacket	2-409
2-87	Contents of ?OPRESP Subpacket	2-411
2-88	Contents of ?OPINFO Subpacket	2-412
2-89	Contents of ?OPEX Main Packet	2-415
2-90	Contents of Access Command Subpacket	2-418
2-91	Contents of Align Command Subpacket	2-419
2-92	Contents of Batch_List Command Subpacket	2-421
2-93	Contents of Batch_Output Command Subpacket	2-422
2-94	Contents of Binary Command Subpacket	2-423
2-95	Contents of Brief Command Subpacket	2-424
2-96	Contents of Cancel Command Subpacket	2-425
2-97	Contents of Consolestatus Command Subpacket	2-428
2-98	Contents of Continue Command Subpacket	2-429
2-99	Contents of CPL Command Subpacket	2-430
2-100	Contents of Create Command Subpacket	2-431
2-101	Contents of Defaultforms Command Subpacket	2-432
2-102	Contents of Disable Command Subpacket	2-434
2-103	Contents of Dismounted Command Subpacket	2-435
2-104	Contents of Elongate Command Subpacket	2-436
2-105	Contents of Enable Command Subpacket	2-438

Table	Caption	
2-106	Contents of Even Command Subpacket	2-439
2-107	Contents of Flush Command Subpacket	2-440
2-108	Contents of Forms Command Subpacket	2-442
2-109	Contents of Halt Command Subpacket	2-443
2-110	Contents of Headers Command Subpacket	2-444
2-111	Contents of Hold Command Subpacket	2-445
2-112	Contents of Limit Command Subpacket	2-446
2-113	Contents of Logging Command Subpacket	2-448
2-114	Contents of LPP Command Subpacket	2-449
2-115	Contents of Mounted Command Subpacket	2-453
2-116	Contents of Mountstatus Command Subpacket	2-455
2-117	Contents of Operator Command Subpacket	2-458
2-118	Contents of Pause Command Subpacket	2-459
2-119	Contents of Premount Command Subpacket	2-460
2-120	Contents of Priority Command Subpacket	2-461
2-121	Contents of Prompts Command Subpacket	2-462
2-122	Contents of Qpriority Command Subpacket	2-465
2-123	Contents of Refused Command Subpacket	2-466
2-124	Contents of Release Command Subpacket	2-467
2-125	Contents of Restart Command Subpacket	2-468
2-126	Contents of Silence Command Subpacket	2-469
2-127	Contents of Spoolstatus Command Subpacket	2-471
2-128	Contents of Stack Command Subpacket	2-473
2-129	Contents of Start Command Subpacket	2-475
2-130	Contents of Status Command Subpacket	2-478
2-131	Contents of Stop Command Subpacket	2-481
2-132	Contents of Trailers Command Subpacket	2-483
2-133	Contents of Unhold Command Subpacket	2-484
2-134	Contents of Unitstatus Command Subpacket	2-486
2-135	Contents of Unlimit Command Subpacket	2-487
2-136	Contents of Unsilence Command Subpacket	2-488
2-137	Contents of Verbose Command Subpacket	2-489
2-138	Contents of Xbias Command Subpacket	2-490
2-139	Contents of Autokey Command Subpacket	2-491
2-140	Contents of Register Command Subpacket	2-492
2-141	Contents of Registerstatus Command Subpacket	2-495
2-142	Contents of ?PCLASS Packet	2-504
2-143	Contents of ?PGNAME Packet	2-509
2-144	Contents of ?PIDS Packet	2-511
2-145	Contents of ?PMTPF Packet	2-513

Table	Caption	
2-146	Contents of ?PRDB/?PWRB Packet	2-520
2-147	?PRDB/?PWRB Packet: Controller Status Words	2-521
2-148	Error Reports Returned in ?PRDB/?PWRB Offsets	2-522
2-149	Contents of ?PROC Packet	2-531
2-150	Privilege Bits in Offset ?PPRV	2-535
2-151	Contents of ?PROC Parameter Packet Extension	2-540
2-152	Contents of ?PROFILE Parameter Packet	2-545
2-153	Contents of ?PSTAT Parameter Packet	2-559
2-154	?PTRDEVICE Function Codes	2-565
2-155	Contents of the ?PTRDEVICE Main Packet	2-567
2-156	Flags for Selecting Pointer Events (Flag Word ?PTRDEV_SET_EVTS.EVTS)	2-569
2-157	Flags for Selecting Buttons (Flag Word ?PTRDEV_SET_EVTS.BTNS)	2-570
2-158	Contents of the ?PTRDEV_SET_DELTA Subpacket	2-571
2-159	Contents of the ?PTRDEV_LAST_EVENT Subpacket	2-572
2-160	Contents of the ?PTRDEV_SET_POINTER Subpacket	2-574
2-161	Contents of the ?PTRDEV_GET_PTR_STATUS Subpacket	2-576
2-162	Contents of the ?PTRDEV_GENERATE_EVENT Subpacket	2-579
2-163	Contents of ?PWDCRYPT Packet	2-582
2-164	Contents of ?RDB/?WRB Packet	2-588
2-165	Contents of ?READ/?WRITE Packet	2-598
2-166	Contents of Screen-Management Packet Extension	2-604
2-167	Contents of Selected Field Translation Packet Extension	2-609
2-168	Contents of ?REGISTER Packet	2-617
2-169	Contents of the ?RNGPR Packet	2-629
2-170	Character Device Characteristics Words	2-645
2-171	Contents of ?SEBL Packet	2-660
2-172	BSC Protocol Data-Link Control Characters (DLCC)	2-662
2-173	Contents of the ?SERMSG Packet	2-672
2-174	Contents of ?SGES Packet	2-676
2-175	Contents of ?SONS Packet	2-687
2-176	Contents of ?SOPPF Packet	2-691
2-177	Contents of ?SPAGE Packet	2-694
2-178	File-Pointer Settings	2-697
2-179	Contents of ?SRCV Packet	2-700
2-180	Masks Returned on ?SRCV System Calls	2-706
2-181	Contents of ?SSND Packet	2-713
2-182	?SSND Call Types	2-717
2-183	Contents of ?SYSPRV Packet	2-735
2-184	Contents of Standard Task Definition Packet	2-738
2-185	Contents of Extended Task Definition Packet	2-742

Table	Caption	
2-186	Contents of Termination Message a 32-bit Process Receives	2-752
2-187	Contents of Termination Message a 16-bit Process Receives	2-756
2-188	?VALIDATE Functions and Their Codes	2-775
2-189	Histogram Array Structure	2-785
2-190	?WINDOW Function Codes	2-788
2-191	Contents of the ?WINDOW Main Packet	2-791
2-192	Contents of the ?WIN_CREATE_WINDOW Subpacket	2-793
2-193	Contents of the ?WIN_DEFINE_PORTS Subpacket	2-798
2-194	Flags in Flag Word ?WIN_SINT.FLAGS	2-803
2-195	Border Types (Offset ?WIN_SINT.BORDER_TYPE)	2-804
2-196	Flags in the ?WIN_GET_USER_INTERFACE Flag Word (Flag Word ?WIN_GINT.FLAGS)	2-805
2-197	Border Types (Offset ?WIN_GINT.BORDER_TYPE)	2-805
2-198	Contents of the ?WIN_GTITLE Subpacket	2-807
2-199	Contents of the ?WIN_WINDOW_STATUS Subpacket	2-810
2-200	Contents of the ?WIN_DEVICE_STATUS Subpacket	2-814
2-201	Contents of ?XINIT Packet	2-830
2-202	Contents of 16-Bit ?XINIT Packet	2-834
2-203	Contents of ?XPSTAT Parameter Packet	2-844
A-1	Error Messages in Order of Octal Code	A-2
A-2	Error Messages in Alphabetical Order	A-15
B-1	Sample Program Sets and their Descriptions	B-2

Licensed Material - Property of Data General Corporation

1

Introduction to System Calls

AOS/VS and AOS/DVS support a wide variety of system calls. System calls are command macros that call on predefined routines in the operating system. The system calls you code into a program allow you to

- Manage processes
- Manage logical address space
- Establish interprocess communications
- Maintain disk files
- Perform file input and output
- Manage a multitasking environment
- Define and access user devices
- Establish customer/server connections between processes
- Perform input and output in blocks, rather than in records or lines
- Register and remove global resources (AOS/DVS only)

You are probably familiar with the Command Line Interpreter (CLI) program. Typically, it accepts statements (i.e., command lines) from a terminal, interprets them, and turns them over to the operating system for execution. An example is the command line

```
) CREATE/DIRECTORY NEW_DIR
```

Here, the CLI calls on the operating system to create a directory named NEW_DIR. More specifically, this CLI command results in an execution of the ?CREATE system call. Your programs can also call on the operating system to perform both functions that the CLI accepts (such as creating a directory file via ?CREATE) and functions that the CLI does not accept (such as getting the next name in a directory file — the function of system call ?GNFN).

Chapter 2 describes each of the AOS/VS and AOS/DVS system calls in alphabetical order. Also, each system call description includes a list of the error codes that are most likely to occur with the particular system call. (See Appendix A for a complete list of the operating systems' error codes.)

All memory addresses are octal and all other numbers are decimal unless otherwise specified.

We assume that you are writing assembly language programs that make system calls. However, you may write high-level language programs that also make direct calls to the operating system. In this latter case, see the general explanation later in this chapter.

Standard Format for System Calls

You must begin each system call with a question mark. Unless otherwise noted, you must reserve two return locations for each system call: a normal return (good return) and an error return (bad return). The standard format for a system call is

?CALL_NAME *[packet address]*
error return
normal return

where:

- *[packet address]* is an optional argument to the system call macro. (See "Parameter Packets" for more explanation.) Some system calls don't have parameter packets. ?DACL is an example.
- error return is a required error return for the system call. (An error return must be a single word instruction.)
- normal return is a required normal return for the system call. (A normal return need not be a single word instruction; that is, a double word instruction would still be a normal return.)

When the operating system executes a system call successfully, it takes the normal return. However, if the system call fails, the operating system takes the error return. It returns an error code in AC0 that tells you why the system call failed.

Frequently, an error code does not indicate an actual error in your program, but rather indicates an exception. For example, you may want to check for error code ERFDE (File Does Not Exist), and then take another action. For simplicity, however, all "exception" codes are called error codes. (See Appendix A for a complete list of the error codes.)

The contents of AC0, AC1, and AC2 after a system call completes depends on the system call you issued. AC3, however, always contains the current frame pointer. AC1, AC2, and AC3 represent accumulator 1, accumulator 2, and accumulator 3, respectively.

Although most system calls conform to the standard format, there are exceptions. The exceptions are noted in the individual system call descriptions. An example is ?RETURN, which does not have a normal return.

Parameter Packets

Some system calls require a *parameter packet* (or simply *packet*). A packet is a set of consecutive words that you set aside in your address space. The operating system uses these words to obtain your input specifications and/or return output values. After you set up a packet, you can either

- Load the packet's address into AC2 before you issue the system call.

or

- Provide the packet address as an argument to the system call macro, which will load the packet address into AC2 for you.

An individual word or double word in a packet is called an *offset* of the packet.

Parametric Coding

Parametric coding means writing your code using mnemonics (symbols listed in PARU.32.SR, PARU_LONG.SR, and PARU.16.SR) to refer to all error codes, packet offsets, and packet values, regardless of how the offsets are ordered in the packet figures.

You should always code parametrically because the packet figures are not true physical representations. Therefore, the exact order of the offsets may vary from one operating system revision to the next. The mnemonics, however, will always correspond to the correct packet offsets. The system call packet in Figure 1–1 is an example of parametric coding.

```
START:  ?OPEN CONSOLE       ;Issue call, with packet address "CONSOLE"
        error return        ; as argument. Supply your own error return
        normal return       ; and good return.

        ;Elsewhere in the program, set up the ?OPEN packet (at the
        ; address specified by "CONSOLE").

CONSOLE:.BLK    ?IBLT               ;Reserve enough words for packet
                                    ; (?IBLT=packet length).

        .LOC    CONSOLE+?ISTI       ;Location of offset ?ISTI, the
                                    ; file specifications word.
        .WORD   ?IEXO!?OFIN         ;Exclusive open, use as input.

        .LOC    CONSOLE+?ISTO       ;Location of offset ?ISTO, the file
                                    ; type.
        .WORD   0                   ;Use defaults for ?ISTO.

        .LOC    CONSOLE+?IBAD       ;Location of offset ?IBAD, a byte
                                    ; pointer to the I/O buffer.
        .DWORD  CBUFFER*2           ; CBUFFER is the console buffer.

        ; Continue coding parametrically for the rest of the packet.
```

Figure 1–1. Parametric Coding Example

We made the packet figures, for each system call requiring a parameter packet, as close as possible to PARU.32.SR, to PARU_LONG.SR, and to PARU.16.SR (for calls with 16-bit packets). We also made the sample packets as close as possible to the packets defined in these parameter files. This arrangement helps you during debugging with the symbolic debugger, since you can use the packet figure to easily match its offsets with the values displayed by the debugger.

Within some of the packet offsets (for example, those in the I/O system calls ?OPEN, ?READ, ?WRITE), you must use bit masks to select options. For example, in Figure 1-1, the specification ?IEXO!?OFIN was formed by applying the OR operator to bit masks ?IEXO and ?OFIN to select two options within the word ?ISTI. In other cases, you may need to set individual bits in a word. The notation for double words is

1S(bit -position)

and, provided that you have specified single words via the MASM macro .ENABLE WORD, the notation for single words is

1B(bit-position)

The S operator generates a 32-bit integer and the B operator generates a 16-bit integer. Thus, the 1S0 notation sets bit 0 (leftmost) of a 32-bit double word to generate 20000000000 and the 1B0 notation sets bit 0 (leftmost) of a 16-bit single word to generate 100000. Other examples are

```
.ENABLE DWORD and 1S1  generates 10000000000
.ENABLE DWORD and 1S2  generates 04000000000
.ENABLE DWORD and 1S29 generates 00000000004
.ENABLE DWORD and 1S31 generates 00000000001
.ENABLE WORD  and 1B1  generates 040000
.ENABLE WORD  and 1B2  generates 020000
.ENABLE WORD  and 1B12 generates 000010
.ENABLE WORD  and 1B15 generates 000001
```

You can also use the B operator with DWORD enabled. Another approach is to use .WORD or .DWORD with respective B or S operators. Two corresponding examples are

```
.WORD   1B15
.DWORD  1S2
```

Other System Call Input

Before you can issue many of the system calls, you must load one or more of the accumulators with input values, such as byte pointers. All accumulators are 32 bits wide under AOS/VS and AOS/DVS, as are all byte pointers.

Each system call description uses unique mnemonics for the high-order and low-order portions of 32-bit values. The high-order portion of a 32-bit value consists of the 16 most significant bits; that is, Bits 0 through 15. The low-order portion of a 32-bit value consists of the 16 least significant bits; that is, Bits 16 through 31.

Under AOS/VS and AOS/DVS, you must define both the high-order and the low-order portions of 32-bit double words. For example, if you pass -1 to indicate the default value of a 32-bit double word, you must set both the high-order and the low-order portions of the word to -1.

There are two methods of setting both the high- and low-order halves of a double word:

1. The preferred method

```
.LOC        CONSOLE+?IBAD
.DWORD  -1
```

   This method sets both the high- and low-order bits to -1 with one easy-to-read MASM pseudo-operation.

2. The other method

```
.LOC    CONSOLE+?IBAD
.WORD   -1                          ; Set high-order bits to -1.

.LOC    CONSOLE+?IBAL
.WORD   -1                          ; Set low-order bits to -1.
```

   This method sets the high- and low-order bits to -1 with separate MASM pseudo-operations.

There is no advantage to using Method 2. Therefore, we advise you to use Method 1. Because Method 1 is simpler than Method 2, this manual does not list low-order packet offsets. The parameter (PARU) files, especially PARU.16, do contain these offsets.

Reserved Symbols

Data General reserves all symbols that begin with a question mark (?) for its internal products, including AOS/VS and AOS/DVS. Don't *define* symbols beginning with ? because assembler errors, unexpected results, or worse can occur. A worst case scenario is your using a symbol that invokes an internal system call which, in turn, alters your program or data in a nonobvious but troublesome way. Of course, you frequently *use* DG-created symbols beginning with ? to invoke system calls and to communicate with the system calls. This manual contains hundreds of such symbols.

A Complete Example — Assembly Language

This section describes an assembly language program that, when executed, creates directory file NEW_DIR and makes it the working directory. The program issues the system calls ?CREATE, which requires a parameter packet, and ?DIR, which does not require a parameter packet.

These system calls have the same effects as the respective CLI commands

```
CREATE/DIRECTORY NEW_DIR
DIR NEW_DIR
```

It's worth noting that your CLI is a process executing a program named CLI.PR. CLI accepts these two character strings, one at a time, and interprets them. Then the CLI loads accumulators, creates any necessary parameter packets, and makes the respective system calls ?CREATE and ?DIR.

The point of this section is to list the general steps of writing assembly language programs that make system calls. This section also contains a comprehensive example from which you can easily generalize.

General Step

Unlike high-level language programs, assembly language requires no special interface to the symbols and values in SYSID.32, PARU.32, PARU_LONG, and PARU.16. Write your program according to the rules of assembly language, and set up system calls according to their documentation in this manual.

Program Listing

Figure 1-2 contains the listing file DIRCREATE.LS that the Macroassembler created from source program file DIRCREATE.SR.

```
SOURCE: DIRCREATE        MASM 06.00.00.00     27-AUG-85 10:12:04  PAGE  1
01                                .TITLE DIRCREATE
02                                .ENT   DIRCREATE
03                                .NREL
04
05                                DIRCREATE:
06 000000 UC 122071 000100        XLEFB  0,NEWDIR*2  ; Byte pointer
07                                                   ;   to new
08                                                   ;   directory's
                                                     ;   name
09 000002 UC 124531               WSUB   1,1         ; 0 in AC1
10                                ?CREATE DIRPKT     ; Create the direc-
11                                                   ;   tory according
12                                                   ;   to its parameter
13                                                   ;   packet.
14 000011 UC 105470               WBR    ERROR       ; Error return
15                                ; Make newly created directory
16                                ;   NEW_DIR the default one.
17 000012 UC 122071 000054        XLEFB  0,NEWDIR*2 ; Byte pointer
18                                                  ;   to new direc-
19                                                  ;   tory's name
20 000014 UC 124531               WSUB   1,1     ; 0 in AC1
21 000015 UC 150531               WSUB   2,2     ; 0 in AC2
22                                ?DIR           ; Move to NEW_DIR.
23 000021 UC 104470               WBR    ERROR
24 000022 UC 150531               WSUB   2,2     ; Execute a normal
25                                ?RETURN        ;   return to AOS/VS.
26 000026 UC 101770               WBR    ERROR
27
28                                DIRPKT:
29 000027 UC  00000000012         .BLK   ?CLTH   ; Packet length
30             00000000027  UC    .LOC   DIRPKT+?CFTYP
31 000027 UC 000012               .WORD  0*400+?FDIR ; Left byte is
                                                     : 0,
32                                                  ;    right specifies
33                                                  ;    a standard (non-
34                                                  ;    control point)
35                                                  ;    directory.
36             00000000030  UC    .LOC   DIRPKT+?CHFS
37 000030 UC 177777               .WORD  -1      ; Hash frame size is
38                                                ;   the default.
```

Figure 1-2. Listing File of Program DIRCREATE.SR (Continues)

```
   SOURCE: DIRCREATE     MASM 06.00.00.00    27-AUG-85 10:12:04  PAGE  2
01              00000000031  UC    .LOC    DIRPKT+?CTIM
02 000031 UC    37777777777        .DWORD  -1      ; Time block is cur-
03                                                 ;   rent date, time.
04              00000000033  UC    .LOC    DIRPKT+?CACP
05 000033 UC    37777777777        .DWORD  -1      ; ACL same as caller's
06              00000000035  UC    .LOC    DIRPKT+?CMSH
07 000035 UC    00000000000        .DWORD  0       ; Set to zero for
08                                                 ;   type ?FDIR.
09              00000000037  UC    .LOC    DIRPKT+?CMIL
10 000037 UC 177777                .WORD   -1      ; Default number of
11                                                 ;   index levels.
12              00000000040  UC    .LOC    DIRPKT+?CMRS
13 000040 UC 000000                .WORD   0       ; Reserved.
14              00000000041  UC    .LOC    DIRPKT+?CLTH ; End of packet
15
16 000041 UC 047105 053537         NEWDIR: .TXT "NEW_DIR" ; Directory name
17           042111 051000
18
19 000045 UC 153211                ERROR:  WLDAI  ?RFEC+?RFCF+?RFER,2 ;Error
20              00000150000
21                                                                     ;return
22                                 ?RETURN
23 000053 UC 175270                WBR     ERROR
24
25                                 .END    DIRCREATE
------------------

XREF:    DIRCREATE        MASM 06.00.00.00     27-AUG-85 10:12:04  PAGE  3
?CACP       00000000004       2/04
?CFTYP      00000000000       1/30
?CHFS       00000000001       1/36
?CLTH       00000000012       1/29    2/14
?CMIL       00000000010       2/09
?CMRS       00000000011       2/12
?CMSH       00000000006       2/06
?CREATE     00000000000 MA    1/10
?CTIM       00000000002       2/01
?DIR        00000000000 MA    1/22
?FDIR       00000000012       1/31
?RETURN     00000000000 MA    1/25    2/22
?RFCF       00000100000       2/19
?RFEC       00000010000       2/19
?RFER       00000040000       2/19
?SYST       00000000000 MA    1/11    1/23    1/26    2/23
?XCALL      00000000001       1/11    1/11    1/23    1/23    1/26    1/26
                              2/23    2/23
DIRCREAT    00000000000 EN    1/02    1/05#   2/25
DIRPKT      00000000027       1/11    1/28#   1/30    1/36    2/01    2/04
                              2/06    2/09    2/12    2/14
ERROR       00000000045       1/14    1/23    1/26    2/19#   2/23
NEWDIR      00000000041       1/06    1/17    2/16#
NO ASSEMBLY ERRORS
```

Figure 1-2. Listing File of Program DIRCREATE.SR (Concluded)

Program Construction

First, be sure you have at least Read access to files MASM.PR, MASM.PS, and LINK.PR. File MASM.PR is the Macroassembler. MASM.PS is a MASM.PR-created file that includes SYSID.32.SR, PARU.32.SR, and PARU_LONG.SR in a special format. Typically, files MASM.PR, MASM.PS, and LINK.PR are in directory :UTIL.

Next are the commands that show the construction of program file DIRCREATE.PR from source file DIRCREATE.SR. All filename suffixes appear for emphasis; you frequently can ignore them during commands to the Macroassembler and Link. The output from the Macroassembler and Link programs doesn't occur in the dialog.

```
) DELETE/2=IGNORE DIRCREATE.LS
) COMMENT Create object file DIRCREATE.OB and list file
) COMMENT  DIRCREATE.LS from DIRCREATE.SR.
) XEQ MASM/L=DIRCREATE.LS DIRCREATE.SR
) COMMENT Create program file DIRCREATE.PR from object file
) COMMENT  DIRCREATE.OB.
) XEQ LINK DIRCREATE.OB
) COMMENT Program file DIRCREATE.PR is now ready for execution.
```

Program Execution

Next is the dialog that shows the execution of program file DIRCREATE.PR. Read it carefully to see what happens when NEW_DIR does not exist prior to DIRCREATE.PR's execution, and then what happens when NEW_DIR does exist prior to DIRCREATE.PR's execution.

```
) COMMENT EXECUTE DIRCREATE.PR after making sure that NEW_DIR
) COMMENT  doesn't exist.
) DELETE/2=IGNORE NEW_DIR
) DIRECTORY
UDD4:ALICE
) DATE; TIME
27-AUG-85
 10:14:40
) XEQ DIRCREATE.PR

) COMMENT There were no errors when ?CREATE and ?DIR executed,
) COMMENT  so no errors were reported.
) FILESTATUS/ASSORT NEW_DIR

DIRECTORY :UDD4:ALICE

  NEW_DIR          DIR   27-AUG-85   10:15:58        0
) DIRECTORY
:UDD4:ALICE
```

```
) COMMENT The ?DIR call in DIRCREATE.PR executed successfully, but
) COMMENT  NEW_DIR was the current directory only during the rest
) COMMENT  of the life of process DIRCREATE.PR. When the process
) COMMENT  terminated, the directory, :UDD4:ALICE, of the parent
) COMMENT  process once again became the current directory.
) DIRECTORY NEW_DIR
) FILESTATUS

) COMMENT As expected, the FILESTATUS command returns no information
) COMMENT    because directory NEW_DIR is empty.
) DIRECTORY ^
) COMMENT We're back in directory ALICE.  Execute DIRCREATE.PR
) COMMENT    again, and see what happens when ?CREATE tries to
) COMMENT    create a file (NEW_DIR) that already exists.
) XEQ DIRCREATE.PR

*ERROR*
FILE NAME ALREADY EXISTS
ERROR: FROM PROGRAM
XEQ,DIRCREATE.PR
) COMMENT Notice how ?RETURN reported the error and terminated
) COMMENT    the process.
```


Licensed Material - Property of Data General Corporation

High-level Language Interface

The previous section explained how assembly language programs can interface with the operating system. High-level language programs can also make system calls. These high-level languages are

- Ada®
- BASIC
- C
- COBOL
- FORTRAN 77
- Pascal
- PL/I

The principle for writing code that makes system calls is the same for each of these languages. For any system call in any language, follow these four steps:

1. Read about the system call in this manual. Note the input/output values of the accumulators and, if any, the parameter packet.
2. Read your language's documentation for its technique for making system calls.
3. In your language, assign values to accumulator variables and to, if any, an array that becomes the parameter packet.
4. In your language, write the statement that makes the system call. The statement includes or refers to the accumulator variables.

For example, suppose you want to create a FORTRAN 77 program that does the same things as assembly language program DIRCREATE.SR. The corresponding four steps in the creation of DIRCREATE.F77 would be as follows.

1. Read the documentation of ?CREATE (with emphasis on the directory option) and ?DIR in this system call dictionary.

2. Read the chapter about the system interface function, ISYS, in the *FORTRAN 77 Environment Manual (AOS/VS)* (093-000288).

3. Use 4-byte integer variables for the values of AC0, AC1, and AC2 when preparing for ?CREATE (whose FORTRAN 77 counterpart is most likely 4-byte integer variable ISYS_CREATE) and for ?DIR (correspondingly ISYS_DIR). Create an array of 2-byte integer variables that is the parameter packet for ?CREATE. Use the BYTEADDR and WORDADDR functions as needed for byte addresses and word addresses, respectively.

4. Use the ISYS function to make the system call. The four arguments to this function are the system call identifier, contents of AC0, contents of AC1, and contents of AC2. The result that ISYS returns is either 0 for no error, or the error code that the operating system placed in AC0.

Appendix B contains FORTRAN 77 program DIRCREATE.F77 that has the same functionality as assembly language program DIRCREATE.SR.

End of Chapter

2

AOS/VS and AOS/DVS System Calls

This chapter contains all AOS/VS and AOS/DVS system calls. The calls appear in alphabetical order.

Most calls apply equally to both operating systems. The classes of exceptions to this statement are

- System calls that apply to AOS/VS only. An example is ?JPSTAT.
- System calls that apply to AOS/DVS only. An example is ?REGISTER.
- System calls that behave differently on the two operating systems. An example is ?PSTAT.

In each of these three cases we indicate that operating system differences occur. In the third case we list the specific differences wherever they occur — accumulators, parameter packet, etc.

The categories of information we supply for each system call follow.

- Name and function.
- Summary of operating system differences (if any).
- Syntax of the assembly language statement call, including error and normal returns.
- Accumulator input and output.
- Error codes.
- Why you might use the call.
- What the call does.

- A figure to show the structure of the parameter packet (if any); this figure is helpful while you are using the debugger and need to examine/change the contents of the packet.
- A table to show the contents of the parameter packet (if any); this table is helpful while you are preparing the contents of the packet during the coding step of program development.
- Notes and references to related documentation (if any). The references are only to other parts of this manual.
- Sample packet (if any).

We frequently use the symbol O/S — for operating system — in the accumulator I/O, figure, and table categories.

Remember — this manual does not explain software concepts, such as the definition of a directory and how you use the CLI program to create and move to directories. Read the *AOS/VS Concepts Manual* and the *AOS/DVS Concepts Manual* for general information about the functionality of system calls. Read this manual for specific information about how to implement these system calls from your programs.

Summary Table

Table 2-1 summarizes all AOS/VS and AOS/DVS system calls, and groups them according to their functions. It also indicates operating system differences. The functions, which appear in Table 2-1 in the following order, are

- AOS/DVS system calls
- Memory management
- Process management
- File creation and management
- File input/output
- Debugging
- Windowing
- Multitasking
- Interprocess communications
- Connection management
- Multiprocessor management

- Class scheduling
- AOS/VS system resources
- User devices
- Bisynchronous communications
- 16-bit processes

There are five columns of information in Table 2-1. Their contents are

- System call name.
- System call description.
- Y to indicate that AOS/VS supports the system call, or nothing to indicate otherwise.
- Y to indicate that AOS/DVS supports the system call, or nothing to indicate otherwise.
- Y to indicate there is a difference in functionality or coding between AOS/VS support of the call and AOS/DVS support of the call, or nothing to indicate otherwise. If the third or fourth column is blank, then the fifth column is also blank.

Licensed Material - Property of Data General Corporation

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

System Call	Function: AOS/DVS System Calls Description	AOS/ VS	AOS/ DVS	OS Diff
?ASSOC/ ?DISSOC	Associates/disassociates two processes for passing obituary messages.		Y	
?DEREGISTER	Deletes an entry from the Public Name Registry.		Y	
?PGLIST	Gets a process group name search list.		Y	
?PGNAME	Returns a process group's identifier.		Y	
?PSLIST	Sets a process group name search list.		Y	
?REGISTER	Creates an entry in the Public Name Registry.		Y	
System Call	**Function: Memory Management** **Description**	**AOS/ VS**	**AOS/ DVS**	**OS Diff**
?ESFF	Flushes shared file memory pages to disk.	Y	Y	
?FLUSH	Flushes the contents of a shared page to disk.	Y	Y	
?GMEM	Returns the number of undedicated memory pages.	Y	Y	
?GSHPT	Lists the current shared partition size.	Y	Y	
?LMAP	Maps a lower ring.	Y	Y	
?MEM	Lists the current unshared memory parameters.	Y	Y	
?MEMI	Changes the number of unshared pages in the logical address space.	Y	Y	
?PMTPF	Permits access to a protected file.	Y	Y	
?RPAGE	Releases a shared page.	Y	Y	
?SCLOSE	Closes a file previously opened for shared access.	Y	Y	
?SOPEN	Opens a file for shared access.	Y	Y	
?SOPPF	Opens a protected shared file.	Y	Y	
?SPAGE	Performs a shared-page read.	Y	Y	
?SSHPT	Establishes a new shared partition.	Y	Y	
?UNWIND	Unwinds the stack and restores the previous environment.	Y	Y	
?VALAD	Validates a logical address.	Y	Y	
?VALIDATE	Validates an area for Read or Write access.	Y	Y	

(Continues)

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

System Call	Function: Process Management Description	AOS/VS	AOS/DVS	OS Diff
?AWIRE	Changes the wiring characteristics of the Agent.	Y	Y	
?BLKPR	Blocks a process.	Y	Y	
?BRKFL	Terminates a process and creates a break file.	Y	Y	
?CHAIN	Passes control from a Ring 7 caller to a new program.	Y	Y	
?CTYPE	Changes a process type.	Y	Y	
?DADID	Gets the PID of a process's father.	Y	Y	
?ENBRK	Enables a break file.	Y	Y	
?EXPO	Sets, clears, or examines execute-protection status.	Y	Y	
?GBIAS	Gets the current bias factor values.	Y	Y	
?GPID	Returns all active PIDs based on a host ID.	Y	Y	
?GUNM	Gets the username of a process.	Y	Y	
?GVPID	Gets the virtual PID of a process.	Y	Y	
?HNAME	Gets a host name or host identifier.	Y	Y	
?KHIST	Kills a histogram.	Y	Y	
?LOCALITY	Changes user locality.	Y		
?MDUMP	Dumps the memory image from a user-specified ring to a file.	Y	Y	
?MPHIST	Starts a histogram on a uni- or multiprocessor system.	Y		
?PCLASS	Get a process's class and locality.	Y		
?PIDS	Gets information about PIDs.	Y	Y	
?PNAME	Gets a full process name.	Y	Y	
?PRIPR	Changes the priority of a process.	Y	Y	
?PROC	Creates a process.	Y	Y	Y
?PROFILE	Performs a profile request.	Y	Y	Y
?PSTAT	Returns status information on a process.	Y	Y	Y
?RESCHED	Reschedules current time slice.	Y	Y	

(Continued)

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

System Call	Function: Process Management Description	AOS/VS	AOS/DVS	OS Diff
?RETURN	Terminates the calling process and passes the termination message to the father.	Y	Y	
?RINGLD	Loads a program file into a specified ring.	Y	Y	
?RNGPR	Returns the .PR filename for a ring.	Y	Y	
?RNGST	Stops ?RINGLD from loading lower rings.	Y	Y	
?RUNTM	Gets runtime statistics on a process.	Y	Y	
?SBIAS	Sets the bias factors.	Y	Y	
?SONS	Gets a list of son processes for a target PID.	Y	Y	
?SUPROC	Enters, leaves, or examines Superprocess mode.	Y	Y	
?SUSER	Enters, leaves, or examines Superuser mode.	Y	Y	
?SYSPRV	Enters, leaves, or examines a process state.	Y	Y	
?TERM	Terminates a process.	Y	Y	
?TPID	Translates a PID.	Y	Y	
?UBLPR	Unblocks a process.	Y	Y	
?UNWIRE	Unwires pages previously wired.	Y	Y	
?WHIST	Starts a histogram.	Y	Y	
?WIRE	Wires pages to the working set.	Y	Y	
?XPSTAT	Returns extended status information on a process.	Y	Y	Y

(Continued)

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

System Call	Function: File Creation and Management Description	AOS/VS	AOS/DVS	OS Diff
?CGNAM	Gets a complete pathname from a channel number.	Y	Y	
?CKVOL	Checks volume identifier of a labeled magnetic tape.	Y	Y	
?CPMAX	Sets maximum size for a control point directory.	Y	Y	
?CREATE	Creates a file or directory.	Y	Y	
?DACL	Sets, clears, or examines a default access control list.	Y	Y	
?DELETE	Deletes a file entry.	Y	Y	
?DIR	Changes the working directory.	Y	Y	
?FLOCK	Locks a file element.	Y	Y	
?FSTAT	Gets file status information.	Y	Y	
?FUNLOCK	Unlocks a file element.	Y	Y	
?GACL	Gets a file entry's access control list.	Y	Y	
?GLINK	Gets the contents of a link entry.	Y	Y	
?GLIST	Gets the contents of a search list.	Y	Y	
?GNAME	Gets a complete pathname.	Y	Y	
?GNFN	Lists a particular directory's entries.	Y	Y	
?GPRNM	Gets a program's pathname.	Y	Y	
?GRNAME	Returns complete pathname of generic file.	Y	Y	
?GTACP	Gets access control privileges.	Y	Y	
?INIT	Initializes a logical disk.	Y	Y	
?MIRROR	Mirrors and synchronizes LDU images.	Y		
?RECREATE	Recreates a file.	Y	Y	
?RELEASE	Releases an initialized logical disk.	Y	Y	
?RENAME	Renames a file.	Y	Y	
?RNAME	Determines whether a pathname contains a reference to a remote host.	Y	Y	
?SACL	Sets a new access control list.	Y	Y	
?SATR	Sets or removes the Permanence attribute for a file or directory.	Y	Y	
?SLIST	Sets the search list for the calling process.	Y	Y	
?SYLOG	Manipulates the system log file.	Y	Y	
?XINIT	Initializes a logical disk (extended).	Y		

(Continued)

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

System Call	Function: File Input/Output Description	AOS/VS	AOS/DVS	OS Diff
?ALLOCATE	Allocates disk blocks.	Y	Y	
?ASSIGN	Assigns a character device to a process.	Y	Y	
?BLKIO	Performs (reads/writes) block I/O.	Y	Y	
?CLOSE	Closes an open channel.	Y	Y	
?CRUDA	Creates a user data area.	Y	Y	
?DEASSIGN	Deassigns a character device.	Y	Y	
?FEOV	Forces end-of-volume on labeled magnetic tape.	Y	Y	
?GCHR	Reads device characteristics of a character device.	Y	Y	
?GCLOSE	Closes a file previously opened for block I/O.	Y	Y	
?GDLM	Gets a delimiter table.	Y	Y	
?GECHR	Get extended characteristics.	Y	Y	
?GOPEN	Opens a file for block I/O.	Y	Y	
?GPOS	Gets the current file-pointer position.	Y	Y	
?GTRUNCATE	Truncates a disk file.	Y	Y	
?INTWT	Defines a console interrupt task.	Y	Y	
?KINTR	Simulates keyboard interrupt sequences.	Y	Y	
?KIOFF	Disables control-character console interrupts.	Y	Y	
?KION	Re-enables control-character console interrupts.	Y	Y	
?KWAIT	Waits for a console interrupt.	Y	Y	
?LABEL	Creates a label for a magnetic tape.	Y	Y	
?ODIS	Disables console interrupts.	Y	Y	
?OEBL	Enables console interrupts.	Y	Y	
?OPEN	Opens a file.	Y	Y	
?PRDB/ ?PWRB	Performs physical block I/O.	Y	Y	
?PWRB	Performs physical block I/O.	Y	Y	
?RDB/?WRB	Performs (reads/writes) block I/O.	Y	Y	
?RDUDA/ ?WRUDA	Reads/writes a user data area.	Y	Y	
?READ/ ?WRITE	Performs (reads/writes) record I/O.	Y	Y	

(Continued)

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

System Call	Function: File Input/Output Description	AOS/ VS	AOS/ DVS	OS Diff
?SCHR	Sets a character device's characteristics.	Y	Y	
?SDLM	Sets a delimiter table.	Y	Y	
?SECHR	Set extended characteristics of a device.	Y	Y	
?SEND	Sends a message to a console.	Y	Y	Y
?SPOS	Sets the position of the file pointer.	Y	Y	
?STOM	Sets the time-out value for a device.	Y	Y	
?TRUNCATE	Truncates a file at the current position.	Y	Y	
?UPDATE	Flushes file descriptor information.	Y	Y	
?WRB	Writes block I/O.	Y	Y	
?WRITE	Writes record I/O.	Y	Y	
?WRUDA	Writes a user data area.	Y	Y	

System Call	Function: Debugging Description	AOS/ VS	AOS/ DVS	OS Diff
?DEBUG	Calls the Debugger utility.	Y	Y	
?GTNAM	Returns symbol closest in value to specified input value.	Y	Y	
?GTSVL	Gets the value of a user symbol.	Y	Y	

(Continued)

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

System Call	Function: Windowing Description	AOS/ VS	AOS/ DVS	OS Diff
?GRAPHICS	Manipulates pixel maps.	Y	Y	
?PTRDEVICE	Controls input from a pointer device.	Y	Y	
?WINDOW	Manipulates windows.	Y	Y	
System Call	**Function: Multitasking Description**	**AOS/ VS**	**AOS/ DVS**	**OS Diff**
?DFRSCH	Disables task rescheduling and indicates prior state of rescheduling.	Y	Y	
?DQTSK	Dequeues one or more previously queued tasks.	Y	Y	
?DRSCH	Disables scheduling.	Y	Y	
?ERSCH	Enables multitask scheduling for the calling process.	Y	Y	
?IDGOTO	Redirects a task's execution path.	Y	Y	
?IDKIL	Kills a task specified by its TID.	Y	Y	
?IDPRI	Changes the priority of a task specified by its TID.	Y	Y	
?IDRDY	Readies a task specified by its TID.	Y	Y	
?IDSUS	Suspends a task specified by its TID.	Y	Y	
?IFPU	Initializes the floating-point unit.	Y	Y	
?IQTSK	Creates a queued task manager.	Y	Y	
?KILAD	Defines a kill-processing routine.	Y	Y	
?KILL	Kills the calling task.	Y	Y	
?MYTID	Gets the priority and TID of the calling task.	Y	Y	
?PRI	Changes the priority of the calling task.	Y	Y	

(Continued)

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

System Call	Function: Multitasking Description	AOS/VS	AOS/DVS	OS Diff
?PRKIL	Kills all tasks of a specified priority.	Y	Y	
?PRRDY	Readies all tasks of a specified priority.	Y	Y	
?PRSUS	Suspends all tasks of a specified priority.	Y	Y	
?REC	Receives an intertask message.	Y	Y	
?RECNW	Receives an intertask message without waiting.	Y	Y	
?SUS	Suspends the calling task.	Y	Y	
?TASK	Initiates one or more tasks.	Y	Y	
?TIDSTAT	Returns status of target task.	Y	Y	
?TLOCK	Protects a task from being redirected.	Y	Y	
?TRCON	Reads a task message from the process console.	Y	Y	
?TUNLOCK	Allows a task to be redirected.	Y	Y	
?UIDSTAT	Returns the status of a task and an unambiguous identifier.	Y	Y	
?WDELAY	Suspends a task for a specified time.	Y	Y	
?XMT	Transmits an intertask message.	Y	Y	
?XMTW	Transmits an intertask message and waits for it to be received.	Y	Y	
?GCPN	Function: Interprocess Communications Gets the console port number.	Y	Y	
?GPORT	Returns the PID associated with a global port number.	Y	Y	
?ILKUP	Returns a global port number.	Y	Y	
?IMERGE	Modifies a ring field within a global port number.	Y	Y	
?IREC	Receives an IPC message.	Y	Y	
?ISEND	Sends an IPC message.	Y	Y	
?ISPLIT	Finds the owner of a port (including its ring number).	Y	Y	
?IS.R	Sends, and then receives an IPC message.	Y	Y	
?TMSG	Defines the termination message format.	Y	Y	
?TPORT	Translates a local port number to its global equivalent.	Y	Y	

(Continued)

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

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

System Call	Function: Multiprocessor Management Description	AOS/ VS	AOS/ DVS	OS Diff
?JPINIT	Initializes a job processor.	Y		
?JPMOV	Moves a job processor to a new logical processor.	Y		
?JPREL	Releases a job processor.	Y		
?JPSTAT	Gets the status of a job processor.	Y		

(Continued)

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

System Call	Function: Class Scheduling Description	AOS/ VS	AOS/ DVS	OS Diff
?CLASS	Gets or sets class IDs.	Y		
?CLSCHED	Enables, disables, or examines class scheduling.	Y		
?CLSTAT	Returns class scheduling statistics.	Y		
?CMATRIX	Gets or sets the class matrix.	Y		
?LPCLASS	Gets/sets logical processor class assignments.	Y		
?LPCREA	Creates a logical processor.	Y		
?LPDELE	Deletes a logical processor.	Y		
?LPSTAT	Gets the status of a logical processor.	Y		
System Call	**Function: AOS/VS System Resources Description**	**AOS/ VS**	**AOS/ DVS**	**OS Diff**
?BNAME	Determines whether process name/queue name is on local or remote remote host.	Y	Y	
?CDAY	Converts a scalar date value.	Y	Y	
?CTOD	Converts a scalar time value.	Y	Y	
?ENQUE	Queues a file entry.	Y	Y	
?ERMSG	Reads the error message file.	Y	Y	
?EXEC	Requests a service from EXEC.	Y	Y	Y
?FDAY	Converts date to a scalar value.	Y	Y	
?FEDFUNC	Interfaces to File Editor utility.	Y	Y	
?FTOD	Converts time of day to a scalar value.	Y	Y	
?GDAY	Gets the current date.	Y	Y	
?GHRZ	Gets the frequency of the system clock.	Y	Y	
?GSID	Gets the system identifier.	Y	Y	
?GTMES	Gets a CLI message.	Y	Y	
?GTOD	Gets the time of day.	Y	Y	
?ITIME	Returns the OS-format internal time.	Y	Y	
?LOGCALLS	Logs system calls.	Y	Y	
?LOGEV	Enters an event in the system log file.	Y	Y	
?OPER	Creates and maintains an operator interface.	Y	Y	Y
?OPEX	Communicates between the current process and an operator process.	Y	Y	Y
?PWDCRYP	Performs a password data encryption request.	Y	Y	
?SDAY	Sets the system calendar.	Y	Y	
?SINFO	Gets selected information about the current operating system.	Y	Y	
?SSID	Sets the system identifier.	Y	Y	
?STOD	Sets the system clock.	Y	Y	

(Continued)

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

System Call	Function: User Devices Description	AOS/ VS	AOS/ DVS	OS Diff
?CLRDV	Clears a device.	Y	Y	
?DDIS	Disables access to all devices.	Y	Y	
?DEBL	Enables access to all devices.	Y	Y	
?IDEF	Defines a user device.	Y	Y	
?IMSG	Receives an interrupt service message.	Y	Y	
?IRMV	Removes a user device.	Y	Y	
?IXIT	Exits from an interrupt service routine.	Y	Y	
?IXMT	Transmits a message from an interrupt service routine.	Y	Y	
?LEFD	Disables LEF mode.	Y	Y	
?LEFE	Enables LEF mode.	Y	Y	
?LEFS	Returns the current LEF mode status.	Y	Y	
?MAPDV	Maps a device into logical address space.	Y	Y	
?STMAP	Sets the data channel map.	Y	Y	

System Call	Function: Bisynchronous Communications Description	AOS/ VS	AOS/ DVS	OS Diff
?SDBL	Disables a BSC line.	Y	Y	
?SDPOL	Defines a polling list or a poll-address/select-address pair.	Y	Y	
?SDRT/ ?SERT	Disables/re-enables a relative console.	Y	Y	
?SEBL	Enables a BSC line.	Y	Y	
?SERT	Re-enables a relative console.	Y	Y	
?SGES	Gets BSC error statistics.	Y	Y	
?SRCV	Receives data or a control sequence over a BSC line.	Y	Y	
?SSND	Sends data or a control sequence over a BSC line.	Y	Y	

(Continued)

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

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

(Concluded)

?ALLOCATE — Allocates disk blocks.

?ALLOCATE *[?RDB/?WRB packet address]*
error return
normal return

Input

Output

AC0	Reserved (Set to 0.)	AC0	Undefined
AC1	Channel number	AC1	Number of bytes allocated (number of blocks * 512)
AC2	Address of the ?RDB/?WRB packet	AC2	Unchanged

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

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.

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. The caller must have Write access to the file. ?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 you supply in the following ?RDB/?WRB packet offsets:

?PSTI	Right byte: number of data blocks to allocate Left byte: ignored
?PRNH (double word)	Starting data block number

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.

?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
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?ASSIGN allows you to link a character device exclusively to one process for the duration of that process. You can issue ?ASSIGN against a device if you want to close it periodically without the risk of losing it to another process.

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 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.

?ASSOC/?DISSOC Associates/Dissociates two processes for passing obituary messages.

AOS/DVS only

?ASSOC *[packet address]*
error return
normal return

?DISSOC *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?ASSOC or ?DISSOC packet

Error Codes in AC0

ERADE	Association does not exist (?DISSOC)
ERPRH	Attempt to access process not in hierarchy (?ASSOC, ?DISSOC)

Why Use It?

Use ?ASSOC to associate the calling process with a target process so that the caller will receive the target process's obituary message.

Here's an EXEC-related use of ?ASSOC. It's possible that MOUNT requests will come to EXEC from processes that are sons of other servers, remote PIDs, etc. EXEC needs to know when these non-son requesting processes terminate so that it can tear down outstanding MOUNT requests. If it doesn't tear down these requests, MOUNT requests remain active and tape drives remain dedicated. After EXEC associates itself with these requesting processes, it receives the necessary obituary messages.

Use ?DISSOC to end the association between two processes that ?ASSOC created.

?ASSOC/?DISSOC Continued

What It Does

If a process issues ?ASSOC, which specifies the VPID (virtual unique pid) of another process, the call establishes a new type of connection. The two processes are, in a sense, attached or "ASSOC"iated. When the target process terminates, the operating system passes its obituary message, via port ?SPTM, to the process that issued ?ASSOC. The issuing process identifies this kind of obituary message because bit ?IFDTH of offset ?ISFL of the obituary message is set. See Figure 2–73.

The termination of the process that issues ?ASSOC does not trigger an obituary message to the target process. The target process is potentially unaware that the association has taken place and may not be equipped to handle the obituary message.

If a process successfully attaches (via ?ASSOC) to its own son, it receives both a normal son termination and an ?ASSOC–type obituary when the son process terminates.

Multiple ?ASSOC calls issued against the same target VPID are treated like multiple ?CON attempts. The operating system reports no "ASSOCIATION ALREADY EXISTS" errors. So, you can issue multiple ?ASSOC calls to ensure an association between two processes.

Also, if the process that is the target of ?ASSOC chains, this does not sever the association. It is only after the chained–to process terminates that the issuing process receives the obituary message. No special "process chained" message goes to the process that issued ?ASSOC.

The ?DISSOC call ends the association between two processes. Note that the termination of the target process does not end an association. So, the issuing process should include a ?DISSOC call as part of its processing of a ?ASSOC type obituary message. (Association functionality is similar to connection management in this regard.)

Both the ?ASSOC and ?DISSOC calls are available only to 32–bit users.

Figure 2–1 shows the structure of the ?ASSOC packet, and Table 2–2 describes its contents. Figure 2–2 shows the structure of the ?DISSOC packet, and Table 2–3 describes its contents.

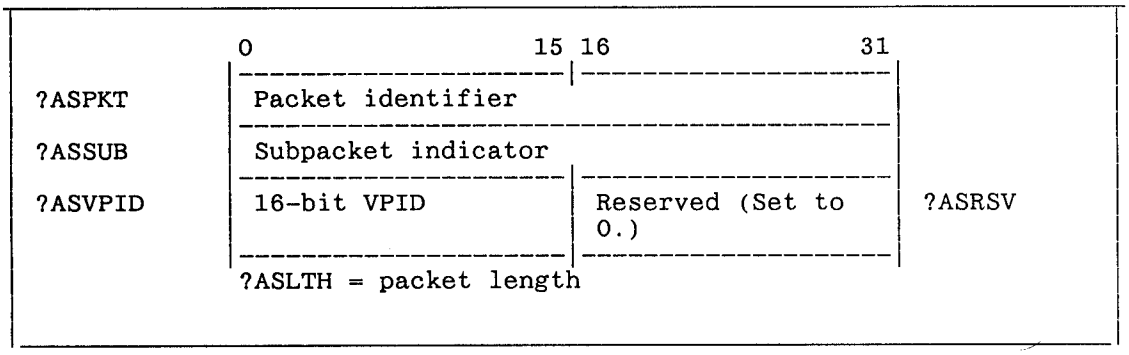


Figure 2–1. Structure of ?ASSOC Packet

Table 2–2. Contents of ?ASSOC Packet

Offset	Contents
?ASPKT (double word)	Packet identifier. Place ?ASID here.
?ASSUB (double word)	Subpacket indicator. Since there is no subpacket, supply 0.
?ASVPID	Supply the 16-bit VPID representation of the target process's UPID. The target process is is the one you want the current (issuing) process to associate with.
?ASRSV	Reserved (Set to 0.)

Licensed Material - Property of Data General Corporation

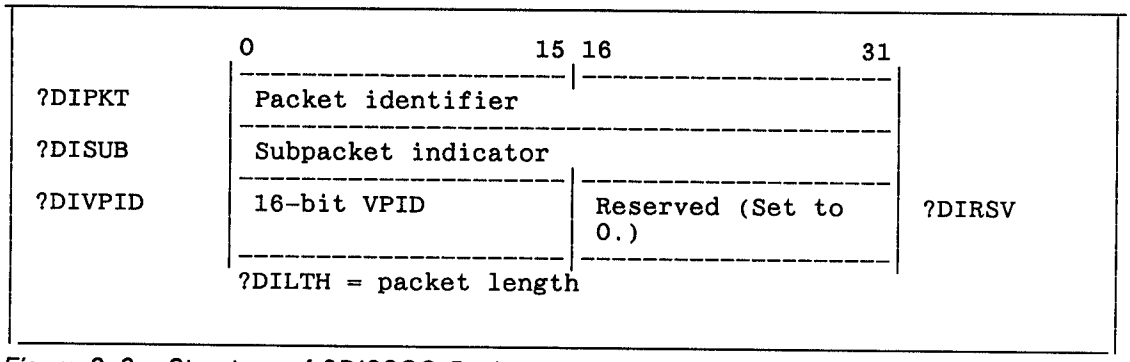


Figure 2–2. Structure of ?DISSOC Packet

Table 2–3. Contents of ?DISSOC Packet

Offset	Contents
?DIPKT (double word)	Packet identifier. Place ?DIID here.
?DISUB (double word)	Subpacket indicator. Since there is no subpacket, supply 0.
?DIVPID	Supply the 16-bit VPID representation of the target process's UPID. The target process is the one you want the current (issuing) process to disassociate from.
?DIRSV	Reserved (Set to 0.)

Example

EXEC could use ?ASSOC and ?DISSOC as follows:

1. Begin processing user requests.
2. Receive a MOUNT request.
3. Map the UPID of the requestor into a 16-bit VPID.
4. Issue ?ASSOC against the requestor's VPID.
5. Allow the user's ?EXEC MOUNT request to return.
6. Process the request.

   ...

X. Receive the obituary message of the ?ASSOC variety.

Y. Tear down the outstanding MOUNT requests associated with the specified VPID.

Z. Issue ?DISSOC against the appropriate VPID.

*** *or* ***

X. Finish processing the user's MOUNT request.

Y. Check to see if there are other outstanding requests under this VPID.

Z. If not, issue ?DISSOC against the user's VPID. If so, go to step 6.

?AWIRE

Changes the wiring characteristics of the Agent.

?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.

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.

Licensed Material - Property of Data General Corporation

?BLKIO — Performs (reads/writes) block I/O.

?BLKIO *[packet address]*
error return
normal return

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

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
ERWAD	Write access denied
ERIOT	Wrong type I/O for open type

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.

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, but 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.)

In addition, ?BLKIO allows you to read the next allocated element in a disk file skipping over unattached blocks. An element is allocated if any of the disk blocks in it have been written to. 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 2-3 shows the structure of the packet, and Table 2-4 describes its contents.

?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.

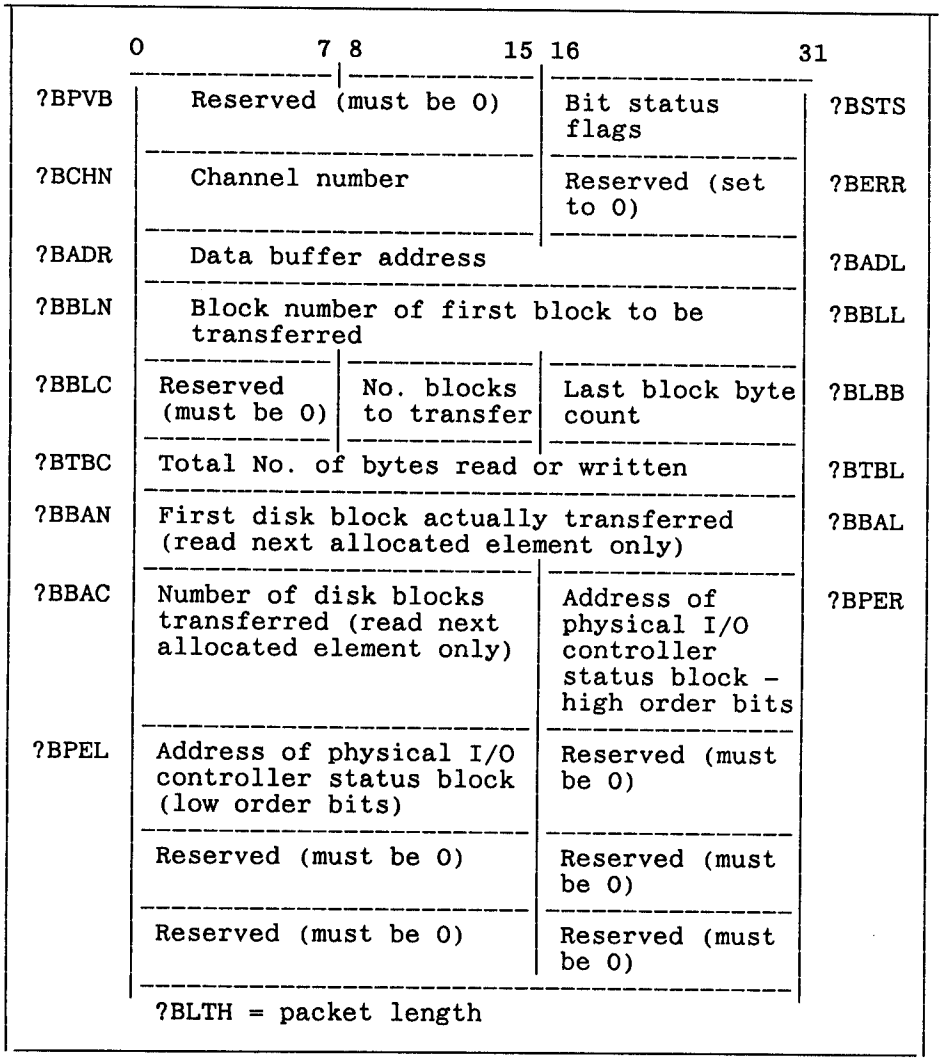


Figure 2-3. Structure of ?BLKIO Packet

Table 2-4. Contents of the ?BLKIO Packet

Offset	Contents
?BPVB	Reserved (must be 0).
?BSTS	To write a block use ?BMIO To perform physical block I/O use ?BMPIO To read the next allocated element use ?BMNAB To perform modified sector I/O use ?BMNMB Other bit status flags are ?BMDIO = transmission mode for MCAs. ?BMEOR = enable VFU load/end of tape override. ?BMAFE = safe write block for magnetic tape.
?BCHN	Channel number (obtained with ?GOPEN).
?BERR	Reserved (must be 0).
?BADR	Address of data buffer in your address space (high-order bits).
?BADL	Address of data buffer (low-order bits).
?BBLN	File block number of first disk block to be transferred (high-order bits); block number or -1 for magnetic tape; link number for MCA.
?BBLL	File block number of first disk block to be transferred (low-order bits); file number for tape; retry count for MCA.
?BBLC	High byte: reserved (must be 0). Low byte: number of blocks to transfer.
?BLBB	Number of bytes in the last disk or tape block transferred; number of bytes in last MCA transmission.

(Continues)

Licensed Material - Property of Data General Corporation

Table 2-4. Contents of the ?BLKIO Packet

Offset	Contents
?BTBC	Total number of bytes read or written (high-order bits).
?BTBL	Total number of bytes read or written (low-order bits).
?BBAN	Block # of first disk block transferred--for read next allocated element only. (high order bits.)
?BBAL	Block # of first disk block transferred (high order bits.)
?BBAC	Number of disk blocks actually transferred--for read next allocated element only.
?BPER	Address of physical I/O controller status block (high-order bits).
?BPEL	Address of physical I/O controller status block (low-order bits).
	Reserved (must be 0).
	Reserved (must be 0).
	Reserved (must be 0).
	Reserved (must be 0).
	Reserved (must be 0).

(Concluded)

Offset ?BSTS must contain a command to indicate the kind of I/O you wish to perform. For example, to write a physical block, you must use the masks ?BMIO and ?BMPIO. To read the next allocated block, just use the mask ?BMNAB.

?BSTS may also include some optional flags for magnetic tapes and MCAs. For details, see ?RDB/?WRB in this chapter.

Offset ?BADR points to the data buffer you reserved in your logical address space for the block I/O transfer. If you read or write disk blocks, use offsets ?BBLN and ?BBLL to indicate the relative block number of the first disk block you want to transfer. (For details of how ?BBLN and ?BBLL are used with tapes and MCAs, see ?RDB/?WRB in this chapter. Note that ?BBLN and ?BBLL correspond to ?PRNH and ?PRNL, respectively.)

If you are writing disk blocks, use offset ?BLBB to indicate the number of bytes in the last block you wish to transfer. If you set this offset to 0, the system sets the bytes in the last block to 512, the default. (For using ?BLBB with tapes or MCAs, see ?RDB/?WRB in this chapter. Note that ?BLBB corresponds to ?PRCL.)

Reading Allocated Blocks

If you use mask ?BMNAB in offset ?BSTS, ?BLKIO reads the next allocated element in a disk file. With this option, the call uses the starting block number that you've indicated in offset ?BBLN and the number of blocks you want transferred (?BBLC), to determine where in the file to begin the read for the next allocated element. This form of the ?BLKIO call returns the file block number of the first block in the next allocated element into offset ?BBAN and the number of blocks it actually transferred into offset ?BBAC and the last byte count in ?BLBB. See Figure 2–4 for examples. The value of ?BBAC is one of the following, depending on which happens first. Blocks will be read until an unallocated block is reached, the number of blocks specified in offset ?BBCL has been satisfied, or the end-of-file is reached, whichever occurs first.

Figure 2–4 gives some examples of how the read next allocated element option works. The illustration assumes 4-block elements (that is, an element size of 4), with slashes indicating that a block has been written to.

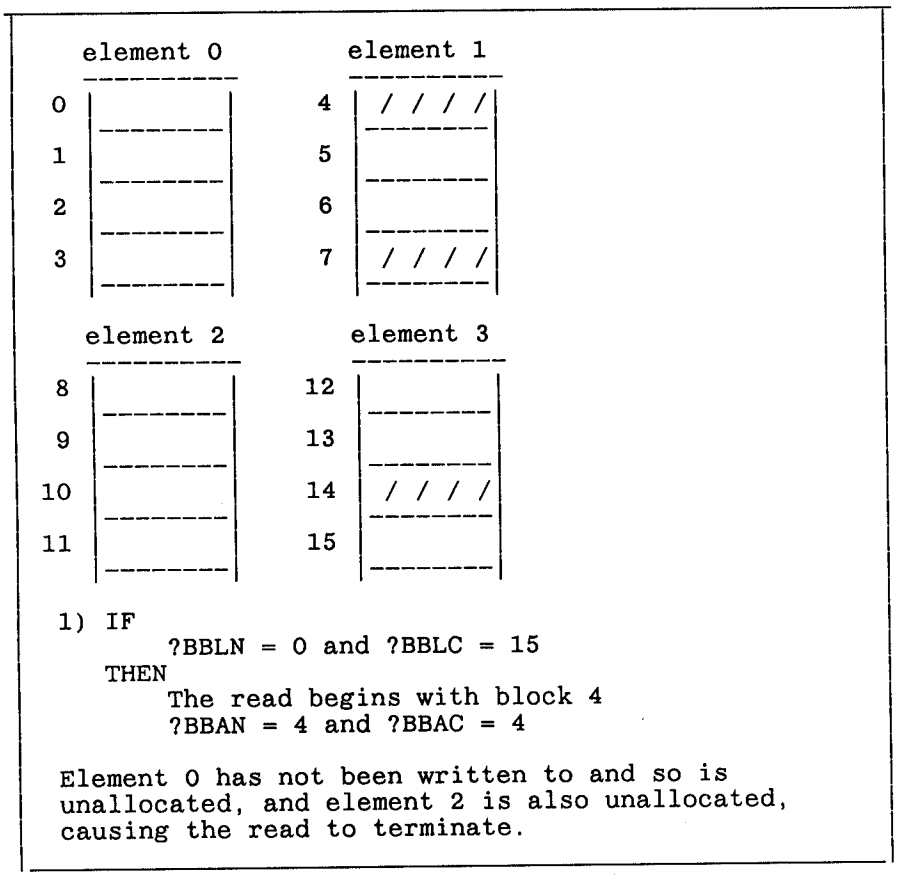


Figure 2–4. Examples of Read Next Allocated Element Option (continues)

```
2) IF
       ?BBLN = 5 and ?BBLC = 15
   THEN
       The read begins with block 5, and
       ?BBAN = 5 and ?BBAC = 3

Element 2 is unallocated, so the read ends. ?BBAN =
5 because, although block 5 has not been written to,
other blocks (4 and 7) in the element have, which
means Element 1 is allocated.

3) IF
       ?BBLN = 7 and ?BBLC = 15
   THEN
       The read begins with block 7, and
       ?BBAN = 7 and ?BBAC = 1

Element 2 is unallocated, so the read ends.

4) IF
       ?BBLN = 2 and ?BBLC = 3
   THEN
       The read begins with block 4, and
       ?BBAN = 4 and ?BBAC = 3

Element 0 is unallocated, so element 1 is used. The
read terminates because the number of blocks
requested (?BBLC or 3) has been reached since it is
less than or equal to the number of blocks (4) in
the element.

5) IF
       ?BBLN = 10 and ?BBLC = 30
   THEN
       The read begins with block 12
       ?BBAN = 12 and ?BBAC = 4

Element 2 is unallocated, so Element 3 is used, but
the file ends before the read can be satisfied, so
EREOF (end of file) is returned.
```

Figure 2–4. Examples of Read Next Allocated Element Option (concluded)

Physical Block I/O

For ?BLKIO to perform physical block I/O on a disk file, you must select ?BMPIO in offset ?BSTS, plus indicate whether you want to write or read (?BMIO to write). In addition, offset ?BPER must point to a block of words that you have reserved for controller status information about the block transfer. The block must be ?BPBLT words long. Figure 2–5 shows the structure of the block.

Licensed Material – Property of Data General Corporation

Into this reserved block, ?BLKIO returns the relative block number of the last block transferred, plus eight controller status words. The words ?BRBB through ?BCS8 are identical to offsets ?PRBB through ?PCS8 in the ?PRDB/?PWRB packet (see Figure 2-5 and Table 2-146 for details). For interpretations of this status block, refer to Table 2-147 and Table 2-148 in the ?PRDB/?PWRB section.

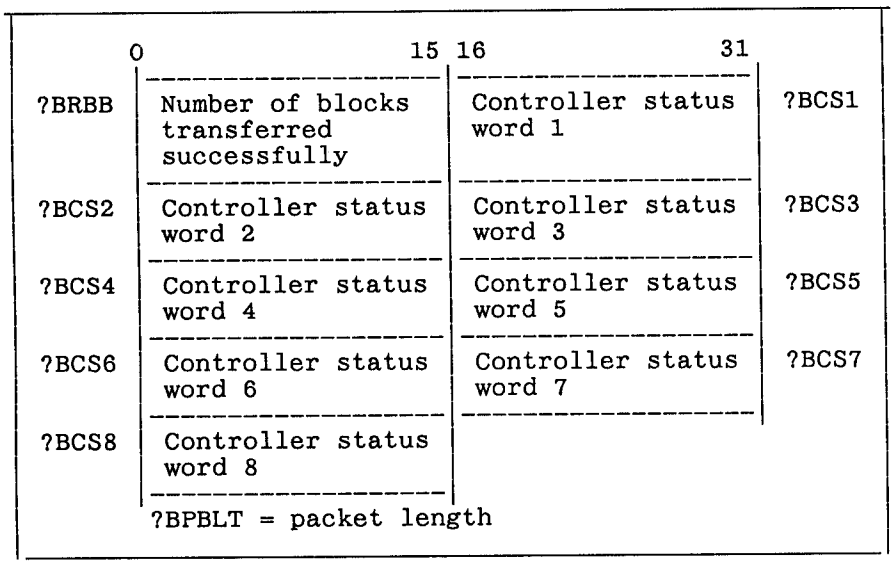


Figure 2-5. Structure of Physical I/O Controller Status Block

For ?BLKIO to perform modified sector I/O to a disk file, you must select ?BMNMB in offset ?BSTS. Since modified sector I/O works in conjunction with physical block I/O, ?BMPIO in offset ?BSTS must also be selected (refer to physical block I/O discussion, above).

When performing modified sector I/O, offset ?BBLC must contain the number of blocks to transfer, plus one block for the modified sector bit map. The first block (256 words) of the transfer contains the modified sector bit map. For more information on modified sector I/O, refer to the manual *Programmer's Reference Series: Models 6236/6237 and Models 6239/6240 Disk Subsystems*.

NOTE: Modified sector I/O is only valid when used on a disk that supports this hardware feature. Currently, this support is only available for 354-megabyte and 592-megabyte "DPJ" disk subsystems.

Notes

- See ?RDB/?WRB and ?PRDB/?PWRB in this chapter.

Licensed Material - Property of Data General Corporation

?BLKPR

Blocks a process.

?BLKPR
error return
normal return

Input

AC0	One of the following: • Byte pointer to the name of the target process • PID of the target process • −1 to block the calling process (This causes the OS to ignore AC1.)
AC1	One of the following: • −1 if AC0 contains a byte pointer • 0 if AC0 contains a PID • AC1 is ignored if AC0 contains −1
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERMPR	System call parameter address error
ERPRE	Invalid system call parameter, AC0 was not −1 and AC1 was not 0 or −1
ERPNM	Illegal process name format
ERPOR	PID is out of range for this process
ERPRH	Attempt to access a process not in hierarchy
ERPRV	You must have Superprocess mode turned on to access another process
ERVBP	Invalid byte pointer to name of target process passed in AC0

Why Use It?

?BLKPR gives you partial control over the system scheduler. You can use ?BLKPR to block one process in favor of another process, or to stop a large job temporarily.

What It Does

?BLKPR allows the calling process to block itself. If the calling process is in Superprocess mode, it can use ?BLKPR to block any process. The calling process can also block direct descendant processes without having Superprocess mode.

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

ERPNM	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.

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, the operating system returns the host ID in Bits 17 through 31 of AC0.

?BRKFL Terminates a process and creates a break file.

?BRKFL
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the name of the target process
- PID of the target process
- −1 to terminate the calling process

AC1 One of the following:

- −1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID
- AC1 is ignored if AC0 contains −1

AC2 One of the following:

- Byte pointer to the break file's pathname
- −1 to use the default break file pathname

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERPRH	Attempt to access a process not in hierarchy
ERVBP	Invalid byte pointer to name of target process passed in AC0

?BRKFL Continued

Why Use It?

?BRKFL is useful when you are debugging a program, because it allows you to save the current state variables of a terminated process.

What It Does

?BRKFL terminates the process that you specify in AC0 (the caller or another process) and creates a break file in the terminated process's working directory. The break file contains the current state of the target process at the time that it terminates. Specifically, the operating system copies the following words to the break file:

Status Word	Contents
BRAC0	Value of AC0
BRAC1	Value of AC1
BRAC2	Value of AC2
BRAC3	Value of AC3
BRPC	Value of the program counter (PC)
BRTID	Task identifier (TID)
BRFP	Value of the stack frame pointer
BRSP	Value of the stack pointer
BRSL	Value of the stack limit
BRSB	Value of the stack base

(Refer to the current AOS/VS Release Notice for more information on the contents of a break file.)

The calling process must have Write or Append access to the target process's working directory. Unless the caller is in Superprocess mode, it cannot issue ?BRKFL against a process that is not its subordinate.

When you issue ?BRKFL, either load AC2 with a byte pointer to the pathname you want to assign to the break file, or use the system-generated default pathname. The default pathname is

?pid.time.BRK

where:

- pid is the 5-digit PID of the target process.
- time is the time of the termination (and break file creation) in the form hours_minutes_seconds.

If the operating system cannot create the break file (for example, if creating a break file causes the working directory to exceed its disk space, or if the caller lacks proper access to the target file's working directory), it takes the error return for ?BRKFL. Note, however, that the target process still terminates.

You can terminate the current process and create a break file by typing CTRL-C CTRL-E at your console. Also, you can direct the operating system to create a break file when the process traps by setting the ?PBRK parameter in the ?PROC packet for a process.

Notes

- See the descriptions of ?PROC and ?MDUMP in this chapter.

?CDAY

Converts a scalar date value.

?CDAY
error return
normal return

Input

AC0	Scalar date you wish to convert (from a base value of 31 December 1967) in Bits 16 through 31
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

utput

AC0	Day from 0 through 31
AC1	Month from 1 through 12
AC2	Year minus 1900 (The result is expressed in octal.)

Error Codes in AC0

No error codes are currently defined.

Why Use It?

Although the system calendar maintains the current date as day, month, and year, the date is sometimes expressed as a scalar value. For example, scalar notation appears in the ?FSTAT packet and in the time block for ?CREATE. Thus, ?CDAY can be a useful conversion tool for both ?FSTAT and ?CREATE.

What It Does

?CDAY converts the scalar value for a date that you specify in AC0 to its equivalent in days, months, and years. The scalar value for a date equals the number of days that have elapsed since 31 December 1967 (the base value).

?CGNAM

Gets a complete pathname from a channel number.

?CGNAM
error return
normal return

Input

AC0	Byte pointer to a receive buffer for the pathname
AC1	Channel number of the target file
AC2	Byte length of the receive buffer

Output

AC0	Unchanged
AC1	Unchanged
AC2	Actual length of the pathname, excluding the terminator

Error Codes in AC0

ERCIU	Channel in use
ERFDE	File does not exist
ERFNO	Channel not open
ERICN	Illegal channel number
ERIRB	Insufficient room in buffer (The buffer that you specified in AC0 is too small to accommodate the pathname.)
ERMPR	System call parameter address error
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

A file's pathname identifies its position in the directory structure. Therefore, you need the pathname as input for many of the file-maintenance system calls. ?CGNAM is a simple tool that allows you to get this information.

What It Does

?CGNAM returns the complete pathname of the file whose channel number you specify in AC1. Before you issue ?CGNAM, set up a receive buffer for the pathname and use a byte pointer in AC0 to address it. In addition, load AC2 with the buffer length, and load AC1 with the target file's channel number. (The operating system returns the target file's channel number in the ?OPEN packet.)

The operating system returns the file's complete pathname, starting with the root, to the receive buffer that you specify.

Notes

- See the description of ?OPEN in this chapter.

Licensed Material - Property of Data General Corporation

?CHAIN

Passes control from a Ring 7 caller to a new program.

```
?CHAIN
error return
```

Input

AC0	Byte pointer to the new program's pathname
AC1	Flag bits (all others = 0): Bit 30 = 1 to flush outstanding IPC messages Bit 30 = 0 to retain (do not flush) outstanding IPC messages Bit 31 = 0 to pass control to the new program's entry point Bit 31 = 1 to pass control to the user debugger
AC2	One of the following: • 0 if there is no IPC message • Address of the IPC message header for the new program, if there is an IPC message

Output

AC0	Undefined
AC1	Undefined
AC2	Undefined

Error Codes in AC0

EREAD	Execute access defined
ERICH	B- or C-type process cannot chain to the target program
ERMEM	Insufficient memory available
ERVBP	Invalid byte pointer to name of target process passed in AC0
ERVWP	Invalid word pointer as a system call argument

Why Use It?

?CHAIN is a way of tying together several steps of a long, complex program set, where each program is a separate segment image. ?CHAIN's main function is to provide compatibility with 16-bit AOS programs, as the 32-bit address space of the MV Family provides room for almost unlimited code and data. In general, try to minimize the use of ?CHAIN, especially in new applications.

In a customer/server relationship, you can use ?CHAIN to tie a customer process to a new program. As long as the connection is from Ring 7, ?CHAIN does not affect the customer process's connection with the server. The operating system sends the server an IPC message to notify it when a customer process issues ?CHAIN. Also, ?CHAIN preserves connections for Ring 3 when a customer chains (provided the connection existed before the ?CHAIN).

Do not issue ?CHAIN while a customer process has requests to servers outstanding. Also, if a customer process chains to a new program, do not issue ?MBFC or ?MBTC against that customer process, unless you can verify the validity of the customer's buffer.

What It Does

?CHAIN must originate from a Ring 7 segment image only. When a Ring 7 process issues ?CHAIN, the operating system releases the resources that the process was using, and then executes a new program. ?CHAIN transfers the following attributes to the new program:

- The username, process name, PID, console, search list, and working directory of the calling process.
- The generic file associations of the calling process (for example, the filenames associated with the generic files @INPUT, @OUTPUT, @LIST, and @DATA).
- The privileges, process type, and priority of the calling process in Ring 7.

?CHAIN does not pass open files, protected shared file grants, or inner rings to the new program. Instead, it closes all files and flushes all unfulfilled I/O. It also removes any user-defined device definitions and destroys inner ring segments. As a result, the ?CHAIN has the effect of terminating all processes created by the inner rings and breaking all connections involving these inner rings.

You can ?CHAIN between processes of different types (i.e., 16-bit processes and 32-bit processes). This is useful if you lack the privilege to create unlimited sons.

Set Bit 31 of AC1 if you want the operating system to assign memory space to the new program, and you want to transfer control to the debugger after the ?CHAIN. Do not set Bit 30 if you want the operating system to pass outstanding IPC messages to the new program. If you want to send a specific IPC message, load AC2 with the address of the IPC header.

The contents of each accumulator is undefined after a ?CHAIN. Because control passes to the new program, ?CHAIN has no normal return.

?CKVOL

Checks volume identifier of a labeled magnetic tape.

?CKVOL
error return
normal return

Input

AC0	Byte pointer to the unit's pathname	
AC1	Byte pointer to the volume identifier (volid) string	
AC2	Bit masks:	
	?XMIBM	if the labels are in IBM format
	?OPDL	if the density mode is 800 bpi
	?OPDM	if the density mode is 1600 bpi
	?OPDH	if the density mode is 6250 bpi
	?OPAM	for automatic density mode selection

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERVOL	Incorrect labeled tape volume mounted

File system error codes

Why Use It?

?CKVOL allows you to make sure the correct labeled tape volume is mounted before you read from or write to the tape.

What It Does

?CKVOL checks the volume identifier (volid) of a labeled magnetic tape against a volid that you (the user) supply.

?CKVOL compares the volid that AC1 points to with the volid of the labeled tape that is mounted on the unit whose pathname AC0 points to. If these two volids are not the same, the operating system returns error code ERVOL in AC0. If the volids are the same, the operating system takes the normal return.

?CLASS — Gets or sets class IDs.

AOS/VS only

?CLASS *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?CLASS packet

Error Codes in AC0

ERXXX	Illegal address
ERCL0	Cannot delete class 0
ERCLU	Class in use
ERIMP	Illegal bit map
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to receive or supply the class IDs that AOS/VS uses to schedule classes.

What It Does

This system call's parameter packet contains a get/set code and a class bit map. The get/set code indicates whether AOS/VS will obtain the class bit map or else will change the class bit map and also change the current configuration of classes. The class bit map identifies which classes are legal; you may then specify them in the class scheduling matrix and LP class hierarchy. There are 16 classes with IDs between 0 and 15, inclusive.

Suppose, in response to a "get" code, AOS/VS sets a bit in the class bit map. This means the corresponding class is a valid one. For example, if AOS/VS has set bits 0, 4, and 5 in the map, then classes with ID values 0, 4, and 5 are valid. Other classes, such as those with ID values 1 and 3, are invalid and will cause errors if you try to use them.

If you want to change the class IDs that AOS/VS is currently using, issue the ?CLASS call with a "set" code. AOS/VS will read the class bit map. It will add/delete classes according to the bits you set. However, you cannot delete a class that appears in the class scheduling matrix; neither can you delete class 0 — the class that exists when AOS/VS first comes up.

Figure 2-6 shows the structure of the ?CLASS parameter packet, and Table 2-5 describes its contents.

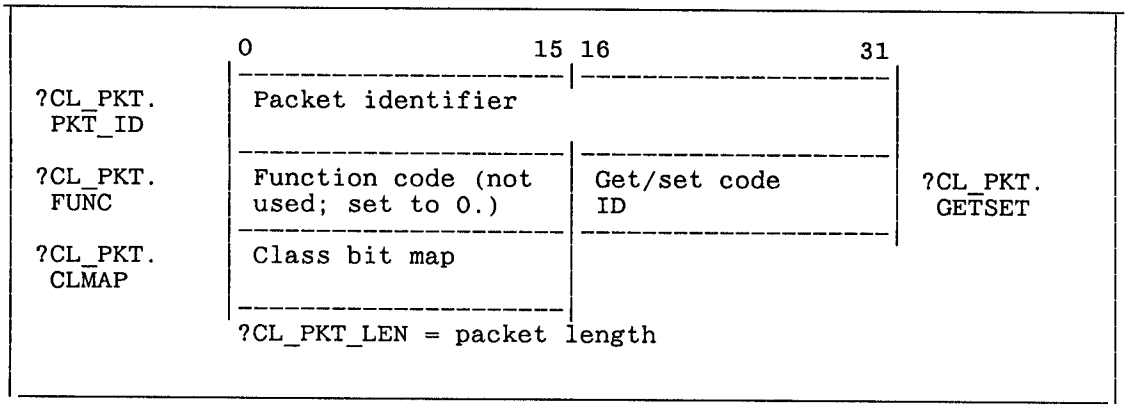


Figure 2-6. Structure of ?CLASS Packet

Table 2-5. Contents of ?CLASS Packet

Offset	Contents
?CL_PKT.PKT_ID (double word)	Packet identifier. Place ?CL_PKT_PKTID here.
?CL_PKT.FUNC	Function code. Not used (set to 0).
?CL_PKT.GETSET	Get/set code. Place the value of ?CL_GET here to obtain the class bit map. Place the value of ?CL_SET here to update the bit map; also place the new bit map in offset ?CL_PKT.CLMAP.
?CL_PKT.CLMAP	The class bit map. You obtain it or set it according to the value in offset ?CL_PKT.GETSET.

?CLOSE — Closes an open channel.

?CLOSE *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?CLOSE packet

Error Codes in AC0

ERACU	Attempt to close unopen channel/device
ERCIU	Channel in use
ERPRE	Invalid system call parameter
ERSIM	Simultaneous requests on same channel
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

After you read from or write to an open I/O channel, issue ?CLOSE to close the channel. This allows the operating system to update the associated file's status information and to free the channel so that it can be used again.

Normally, if a process terminates, the operating system automatically closes all its open channels. However, if a process terminates abnormally (for example, because of a CTRL-C CTRL-B sequence or a user trap), the operating system flushes all buffers of any open channels before it closes the channels.

When you issue ?CLOSE against a labeled tape file, the operating system writes the user trailer labels from the area that you specify in offset ?ELUT of the ?OPEN packet extension.

What It Does

?CLOSE closes an open channel and frees the channel number that the operating system assigned to that channel.

?CLOSE requires the I/O packet shown in Figure 2-7. You can specify the address of the packet as an argument to ?CLOSE, or you can load AC2 with a byte pointer to the packet address before you issue ?CLOSE.

?CLOSE Continued

As Figure 2–7 shows, ?CLOSE uses the same I/O packet as ?OPEN, ?READ, and ?WRITE. Only one I/O parameter applies to ?CLOSE:

- Channel Number, offset ?ICH, which is the number of the channel you want to close. (The operating system returns the channel number to offset ?ICH after the ?OPEN.)

Also, there are no extended packets for ?CLOSE.

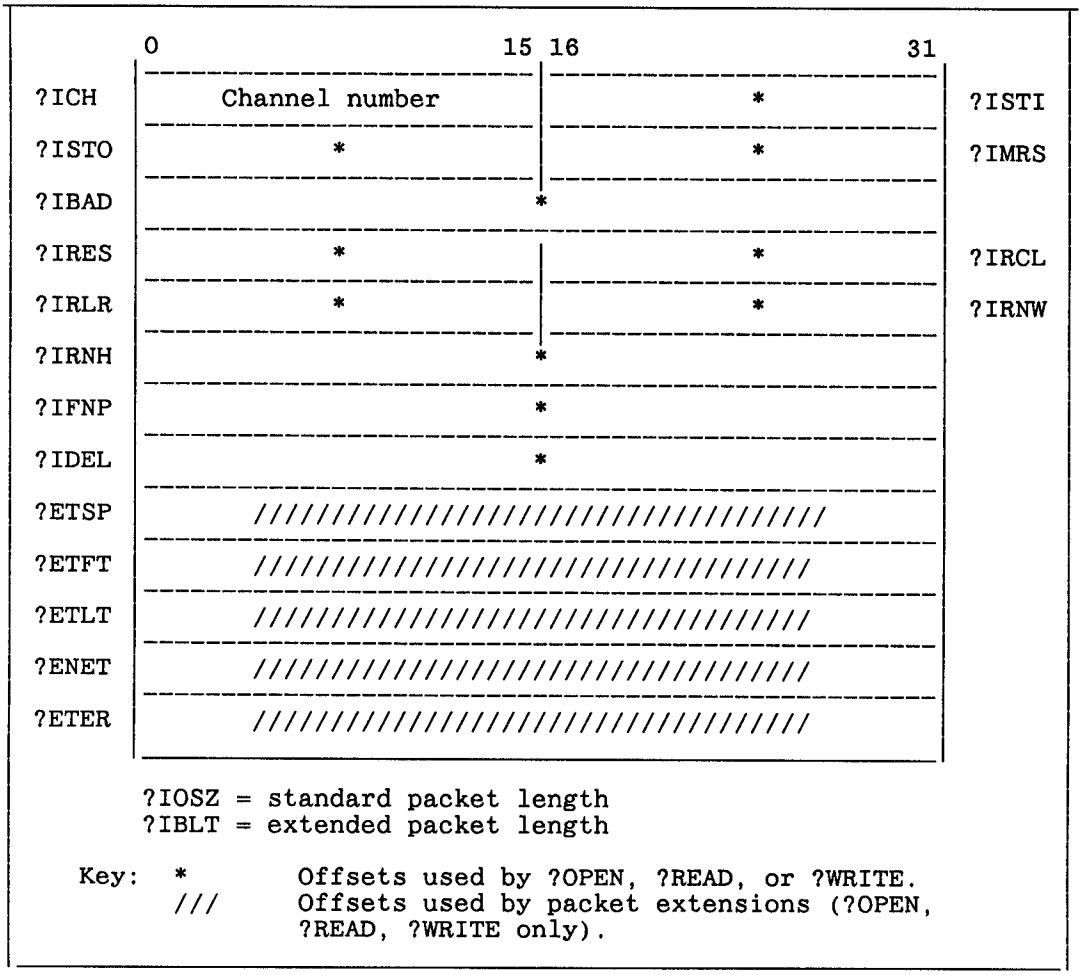


Figure 2–7. Structure of ?CLOSE Packet

Notes

- See the descriptions of ?OPEN, ?READ, and ?WRITE in this chapter for more information on I/O parameters.

?CLRDV

Clears a device.

?CLRDV
error return
normal return

Input

AC0 contains either:

- Byte pointer to the console name
- Channel (number) on which the device is opened

AC1 contains the following:

Bit 0 = 0 if AC0 contains a byte pointer

Bit 0 = 1 if AC0 contains a channel number

AC2 option:

- ?CDRXON — receive XON (simulate CTRL-Q)
- ?CDSBRK — send a break

Output

AC0 unchanged

AC1 unchanged

AC2 unchanged

Error Codes in AC0

ERFDE	File does not exist
ERICN	Illegal channel
ERIFD	Illegal function for device (Returned when trying to transmit a break to a device other than an IAC, or receiving XON on an input only device.)
ERIFT	Illegal file type
ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action
ERVBP	Invalid byte pointer passed as system call argument

Licensed Material - Property of Data General Corporation

?CLRDV Continued

Why Use It?

The receive XON option is useful in restarting output on a line which cannot be restarted by any of the typical methods (i.e., CTRL–Q, hold, etc.). For example, if a line is in binary mode, it will not recognize control sequences, thus CTRL–Q will not be interpreted as an XON command. It also is useful in restarting output on asynchronous lines that do not have consoles. These lines may be held on CTRL–S from characters generated by line noise.

The send break option offers a simple method of sending a break to an IAC line.

What It Does

?CLRDV allows you two options: to simulate an XON or to transmit a break to an IAC line. The first option, ?CDRXON, allows you to restart output by simulating receiving an XON. This function is also accomplished by typing a control–Q, but ?CLRDV allows you to do this from a remote device running either PID 2 or a process which owns the target device. The second option, ?CDSBRK, allows you to send a break to an IAC device. This option, also, can only be made by PID 2 or the owner of the target device.

?CLSCHED

Enables, disables, or examines class scheduling.

AOS/VS only

?CLSCHED *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?CLSCHED packet

Error Codes in AC0

ERPRV	Caller not privileged for this action
ERCSO	Class scheduling is enabled

Why Use It?

Use this system call to enable or disable class scheduling for a specified set of LPs. You may also accumulate class scheduling statistics.

What It Does

You implement this system call in one of two basic ways: enable class scheduling with accumulation of statistics, or simply accumulate statistics about class scheduling with class scheduling disabled. You cannot enable class scheduling and disable accumulation of statistics.

For class scheduling — use this call to turn class scheduling on or off according to logical processors you specify. When an LP's class scheduling is on, AOS/VS uses the LP's class hierarchy and time interval allotments to select processes. When an LP's class scheduling is off, AOS/VS ignores the LP's class hierarchy and time interval allotments; AOS/VS also eliminates certain scheduling and accounting overhead.

For statistical accumulation — use this call to accumulate statistics that show how AOS/VS is allotting processor time. This accumulation occurs according to logical processors you specify. It always occurs when class scheduling is enabled and can occur without class scheduling if desired. Use the ?CLSTAT call to view the accumulated statistics.

?CLSCHED Continued

Use the ?CLSCHED packet entries as follows:

- Use the get/set offset (?CLS_PKT.GETSET) to indicate whether you want to obtain the system class schedule settings or to set them. Place ?CLS_GET in the offset in the former case or ?CLS_SET in the offset in the latter case.

- Use the accumulate/schedule offset (?CLS_PKT.FLAVOR) to indicate which LP setting you wish to get/set: statistical accumulation or class scheduling.

- Offset ?CLS_PKT.LPMAP contains an LP bit map. Bit 0 corresponds to LPID 0, etc. When using the "set" code, a 1 bit tells AOS/VS to turn class scheduling or accumulation on for the corresponding LP; a 0 bit tells AOS/VS to turn class scheduling or accumulation off for that LP. AOS/VS does *not* return an error if an LP is already in the desired scheduling or accumulation mode.

  Similarly, if you enter the "get" code, AOS/VS uses the LP bit map to tell you which LPs have class scheduling or accumulation enabled.

Note the following rules that apply to enabling/disabling class scheduling and statistical accumulation:

- Enabling class scheduling for an LP has AOS/VS automatically turn on statistical accumulation for the LP.

- Disabling class scheduling for an LP has AOS/VS automatically turn off statistical accumulation for the LP.

- However, you may turn on statistical accumulation for an LP, regardless of whether you have enabled or disabled class scheduling.

- You *cannot* disable statistical accumulation if the LP has class scheduling enabled.

Given these rules, the easiest way to configure your LP scheduling and accumulation is to

- Issue the scheduling version of ?CLSCHED first.

- Then issue the accumulation version of ?CLSCHED.

Figure 2-8 shows the structure of the ?CLSCHED parameter packet, and Table 2-6 describes its contents.

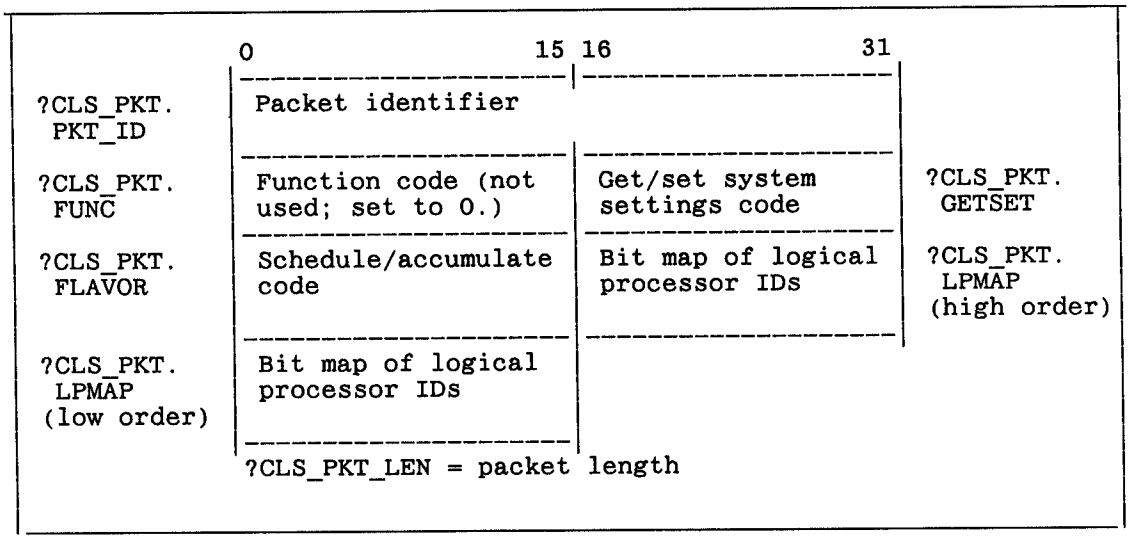


Figure 2-8. Structure of ?CLSCHED Packet

Table 2-6. Contents of ?CLSCHED Packet

Offset	Contents
?CLS_PKT.PKT_ID (double word)	Packet identifier. Place ?CLS_PKT_PKTID here.
?CLS_PKT.FUNC	Function code. Not used (set to 0).
?CLS_PKT.GETSET	Code word into which you place ?CLS_GET when you want to obtain system settings and ?CLS_SET when you want to set system settings.
?CLS_PKT.FLAVOR	Code word into which you place ?CLS_ACC when you want to get/set LP statistical accumulation or ?CLS_SCHED when you want to get/set LP class scheduling. You can specify both values when you want to find out which logical processors are accumulating statistics.
?CLS_PKT.LPMAP (double word)	Bit map of the logical processors, with bit 0 corresponding to LP 0, etc.

?CLSCHED Continued

Notes

- When you initially create an LP, class scheduling and statistics accumulation are disabled.
- AOS/VS maintains scheduling statistics from the time that it creates an LP. If you turn statistics accumulation off and on, the statistics will not include information about the processes that were scheduled when the accumulation was off.
- AOS/VS returns an error if you don't have the appropriate privilege (System Manager) to issue this call.
- AOS/VS continues to maintain LP class hierarchies, class IDs, and the class matrix whether or not class scheduling is enabled. This means ?LPCLASS, ?CLASS, and ?CMATRIX calls work correctly whether or not class scheduling is in use.

?CLSTAT — Returns class scheduling statistics.

AOS/VS only

?CLSTAT *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?CLSTAT packet

Error Codes in AC0

ERICD	Illegal function code
ERLNE	LP does not exist
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to obtain information about the allocation of a logical processor's CPU time.

Licensed Material - Property of Data General Corporation

?CLSTAT Continued

What It Does

The ?CLSTAT call supports a main packet and an optional subpacket. The main packet provides information about the division of an LP's CPU time among the user community. The subpacket indicates how CPU time is used among various portions of AOS/VS. You tell AOS/VS whether or not to place information in the subpacket by the value you place in offset ?CLST_PKT.FUNC of the main packet.

Many of the offsets are 4 words long and contain a time value. Here is the format of the time value:

- It's 64 bits long.
- Its frequency is (2**18)*(10**4) = 2,621,440,000 cycles per second.
- Bit number n (n is between 0 and 63) changes every (2**(63–n))/((2**18)*(10**4)) seconds. So, for example:
  - Bit 45 set and the others off represents 100 microseconds.
  - Bit 31 set and the others off represents 1.6384 seconds.
  - Bit 20 set and the others off represents 3,355 seconds.
  - Bit 6 set and the others off represents about 1 year, 9 months.
- It rolls over every 223 years.

Figure 2–9 shows the structure of the ?CLSTAT main packet, and Table 2–7 describes its contents. Figure 2–10 shows the structure of the ?CLSTAT subpacket, and Table 2–8 describes its contents.

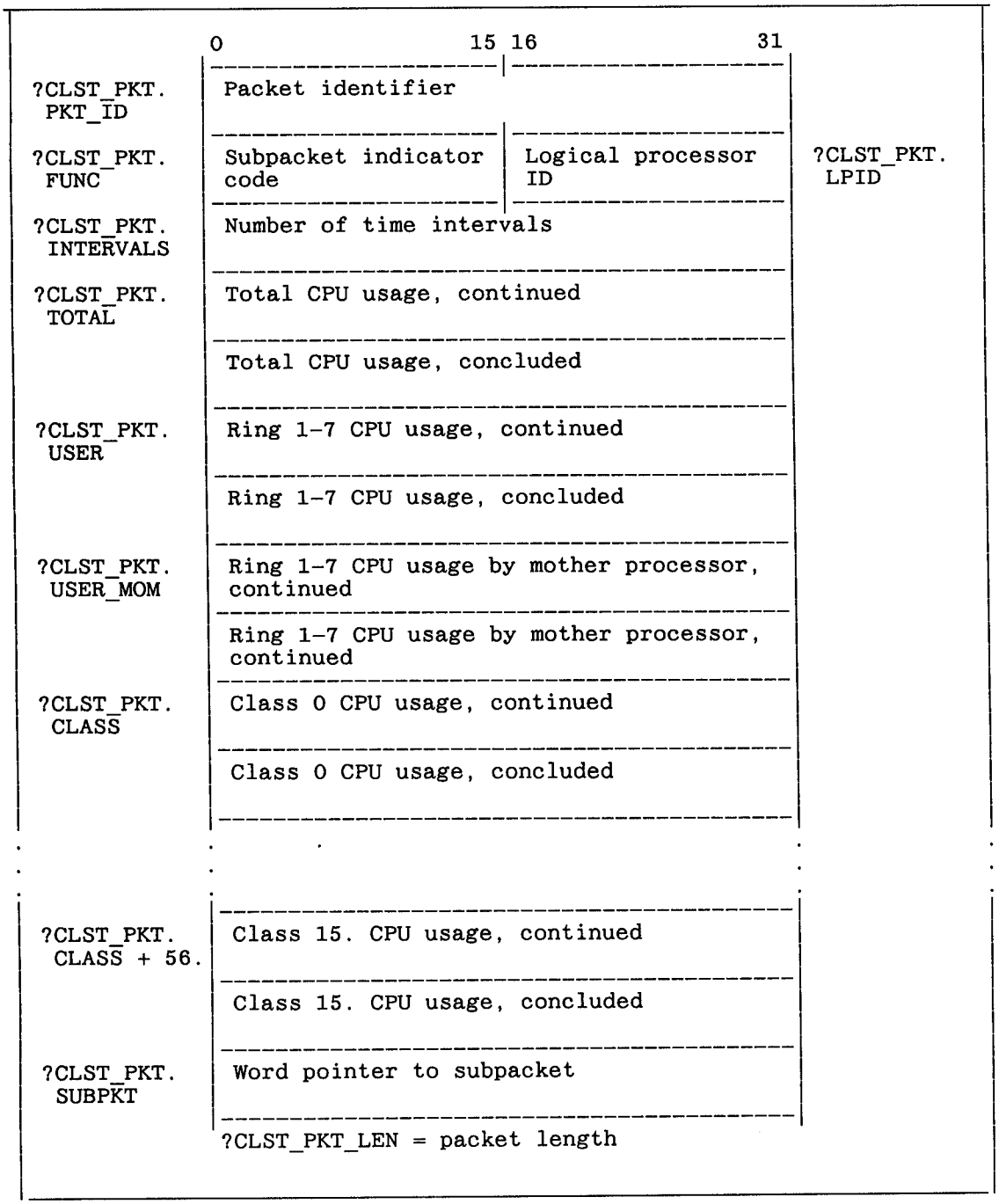


Figure 2–9. Structure of ?CLSTAT Main Packet

Licensed Material – Property of Data General Corporation

?CLSTAT Continued

Table 2-7. Contents of ?CLSTAT Main Packet

Offset	Contents
?CLST_PKT.PKT_ID (double word)	Packet identifier. Place ?CLST_PKT_PKTID here.
?CLST_PKT.FUNC	Subpacket indicator code. Place ?CLST_AOSVS here to indicate the presence of a subpacket or ?CLST_NONE to indicate its absence.
?CLST_PKT.LPID	Logical processor ID. Place the ID of the LP, about which you want class scheduling statistics, here.
?CLST_PKT.INTERVALS (double word)	AOS/VS returns the number of scheduling time intervals that have elapsed.
?CLST_PKT.TOTAL (four words)	AOS/VS returns the total amount of CPU time that the logical processor (i.e., its job processors) has used.
?CLST_PKT.USER (four words)	AOS/VS returns the total amount of CPU time that the logical processor has used in rings 1 through 7.
?CLST_PKT.USER_MOM (four words)	AOS/VS returns the total amount of CPU time that the logical processor has used in rings 1 through 7 on the mother processor (i.e., the LP couldn't run on a child processor). This value is included in offset ?CLST_PKT.USER.
?CLST_PKT.CLASS (four words)	AOS/VS returns the total amount of CPU time that scheduling class number 0 used.
.	.
.	.
.	.
?CLST_PKT.CLASS+56. (four words)	AOS/VS returns the total amount of CPU time that scheduling class number 15. has used.
?CLST_PKT.SUBPKT (double word)	If you specified a subpacket in offset ?CLST_PKT.FUNC, then place its word address here.

Licensed Material - Property of Data General Corporation

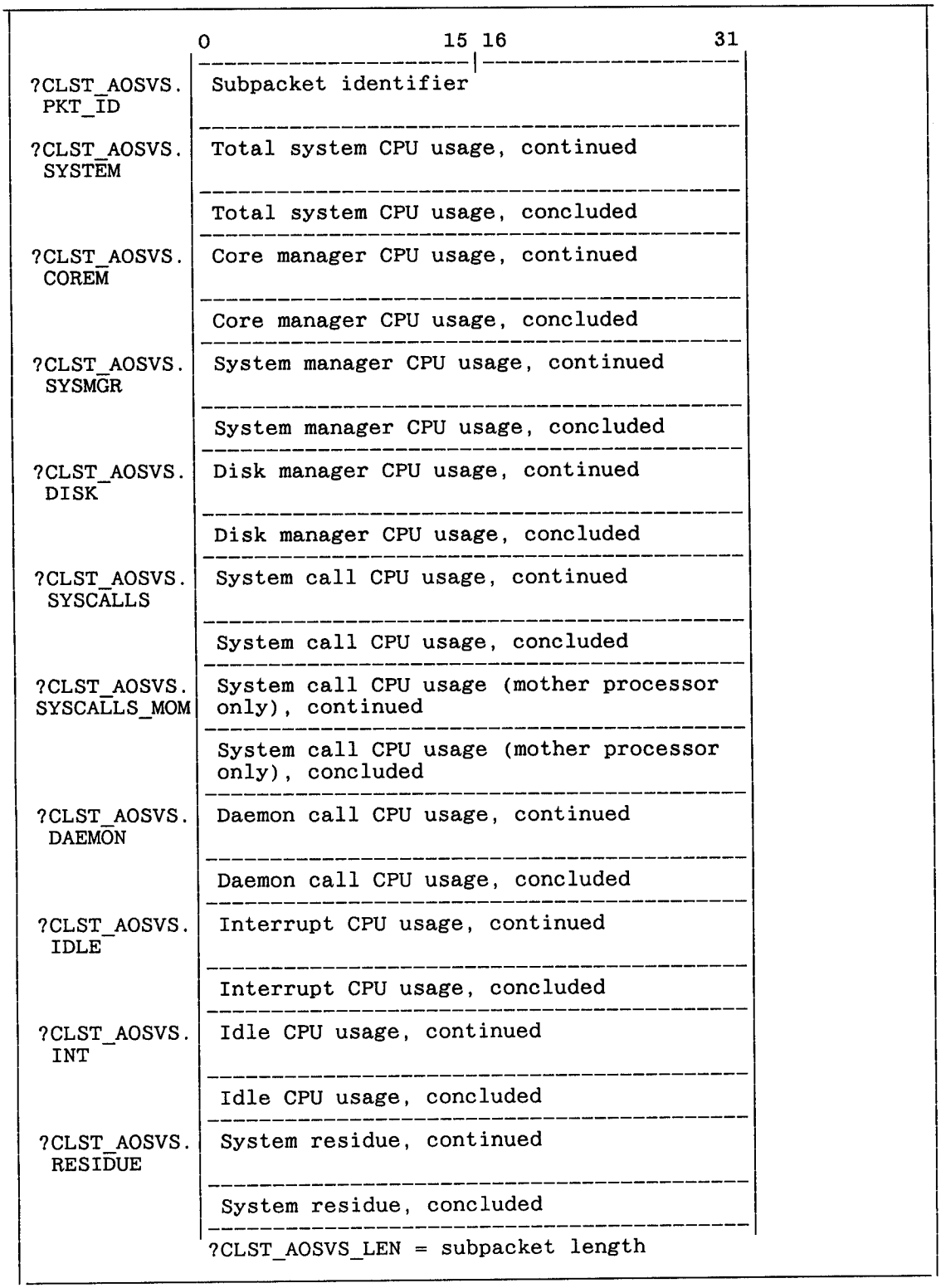


Figure 2–10. Structure of ?CLSTAT Subpacket

?CLSTAT Continued

Table 2-8. Contents of ?CLSTAT Subpacket

Offset	Contents
?CLST_AOSVS.PKT_ID (double word)	Subpacket identifier. Place the value of ?CLST_AOSVS_PKTID here.
?CLST_AOSVS.SYSTEM (four words)	AOS/VS returns the amount of total system CPU usage.
CLST_AOSVS.COREM (four words)	AOS/VS returns the amount of core memory manager CPU usage.
CLST_AOSVS.SYSMGR (four words)	AOS/VS returns the amount of system manager CPU usage.
CLST_AOSVS.DISK (four words)	AOS/VS returns the amount of disk manager CPU usage.
CLST_AOSVS.SYSCALLS (four words)	AOS/VS returns the amount of system call CPU usage.
CLST_AOSVS. SYSCALLS_MOM (four words)	AOS/VS returns the amount of system call CPU usage (mother processor only).
CLST_AOSVS.DAEMON (four words)	AOS/VS returns the amount of daemon call CPU usage.
CLST_AOSVS.IDLE (four words)	AOS/VS returns the amount of interrupt CPU usage.
CLST_AOSVS.INT (four words)	AOS/VS returns the amount of idle CPU usage.
CLST_AOSVS.RESIDUE (four words)	AOS/VS returns the amount of system residue time.

?CMATRIX — Gets or sets the class matrix.

AOS/VS only

?CMATRIX *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?CMATRIX packet

Error Codes in AC0

ERICI	Illegal class ID
ERILV	Illegal locality value
ERPRV	Caller not privileged for this action

Why Use It?

Use ?CMATRIX to receive the current class locality scheduling matrix from AOS/VS or to modify this matrix.

What It Does

The following list describes the format of the class locality scheduling matrix in ?CMATRIX's subpacket.

- The matrix consists of 256 cells, with 2 words in each cell. Each cell contains a user locality, a program locality, and the corresponding class ID.
- User locality values and program locality values appear consecutively in the 2 bytes of the first word in each cell. Both values are between 0 and 15 inclusive.
- The class ID corresponding to a user/program locality value is in the word following the locality value. A class ID value is between 0 and 15 inclusive.

?CMATRIX Continued

Specifying the "get" code in the main packet has AOS/VS return the class matrix in the subpacket. Specifying the "set" code in the main packet and supplying a class matrix in the subpacket has AOS/VS update its scheduling practices for the indicated users, programs, and classes. However — this updating does not affect processes that are currently running; their classes are *not* modified. When AOS/VS creates processes after ?CMATRIX it gives them their class values according to the new matrix. Use ?LOCALITY to change the class of a process that is currently running.

When you issue ?CMATRIX, indicate as many or as few cells of the matrix as you wish. Here are the rules:

- Place the number of cells (2-word sets) you want to access in the cell count offset of the main packet.
- Supply that many sets of user locality, program locality, and class ID values in the subpacket.
- If you use the "set" version of the call, supply the desired class ID values in the matrix.
- If you use the "set" version of the call and supply duplicate specifications (i.e., multiple values for the same cell), AOS/VS will use the last class ID value given for the cell.
- If you use the "get" version of the call, AOS/VS returns the class IDs associated with the user/program locality pairs that you specified.

Figure 2–11 shows the structure of the ?CMATRIX main packet, and Table 2–9 describes its contents. Figure 2–12 shows the structure of the ?CMATRIX subpacket, and Table 2–10 describes its contents.

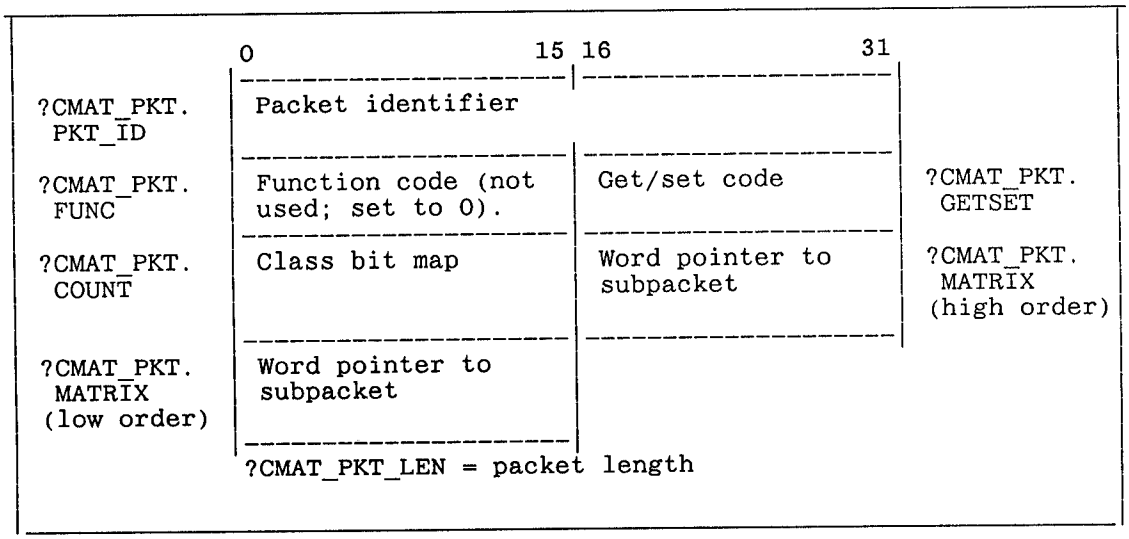


Figure 2–11. Structure of ?CMATRIX Main Packet

Table 2–9. Contents of ?CMATRIX Main Packet

Offset	Contents
?CMAT_PKT.PKT_ID (double word)	Packet identifier. Place ?CMAT_PKT_PKTID here.
?CMAT_PKT.FUNC	Function code. Not used (set to 0).
?CMAT_PKT.GETSET	Get/set code. Place the value of ?CMAT_GET here to obtain the class scheduling matrix in the subpacket. Place the value of ?CMAT_SET here to update the matrix; also place the matrix's new values in the subpacket.
?CMAT_PKT.COUNT	The number of cells (2 words in each cell) in the subpacket.
?CMAT_PKT.MATRIX	Word address of the subpacket.

Licensed Material – Property of Data General Corporation

?CMATRIX Continued

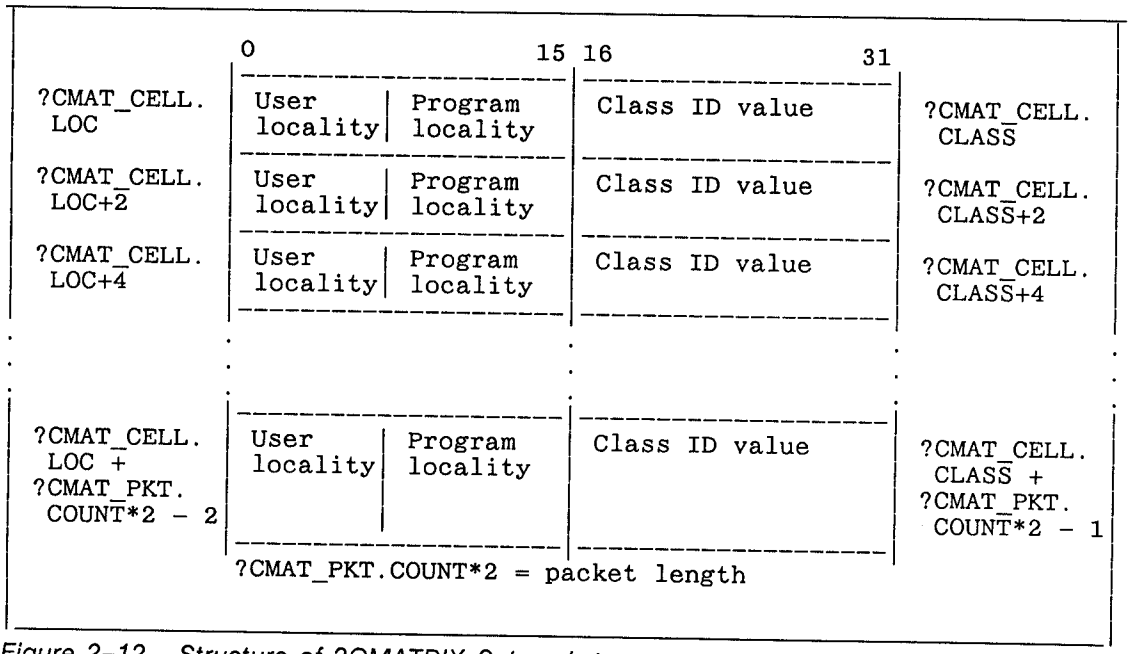


Figure 2-12. Structure of ?CMATRIX Subpacket

Table 2-10. Contents of ?CMATRIX Subpacket

Offset	Contents
?CMAT_CELL.LOC	User locality (left byte) and program locality (right byte) pair for the first cell.
?CMAT_CELL.CLASS	Class ID of the first cell.
?CMAT_CELL.LOC+2	User locality (left byte) and program locality (right byte) pair for the second cell.
?CMAT_CELL.CLASS+2	Class ID of the second cell.
?CMAT_CELL.LOC+4	User locality (left byte) and program locality (right byte) pair for the third cell.
?CMAT_CELL.CLASS+4	Class ID of the third cell.
.	.
.	.
.	.
?CMAT_CELL.LOC + ?CMAT_PKT.COUNT*2-2	User locality (left byte) and program locality (right byte) pair for the last cell.
?CMAT_CELL.CLASS + ?CMAT_PKT.COUNT*2-2	Class ID of the last cell.

Licensed Material - Property of Data General Corporation

It's possible to look at the offsets in Table 2–10 another way. Figure 2–13 shows a relation between the ?CMATRIX main packet and its subpacket.

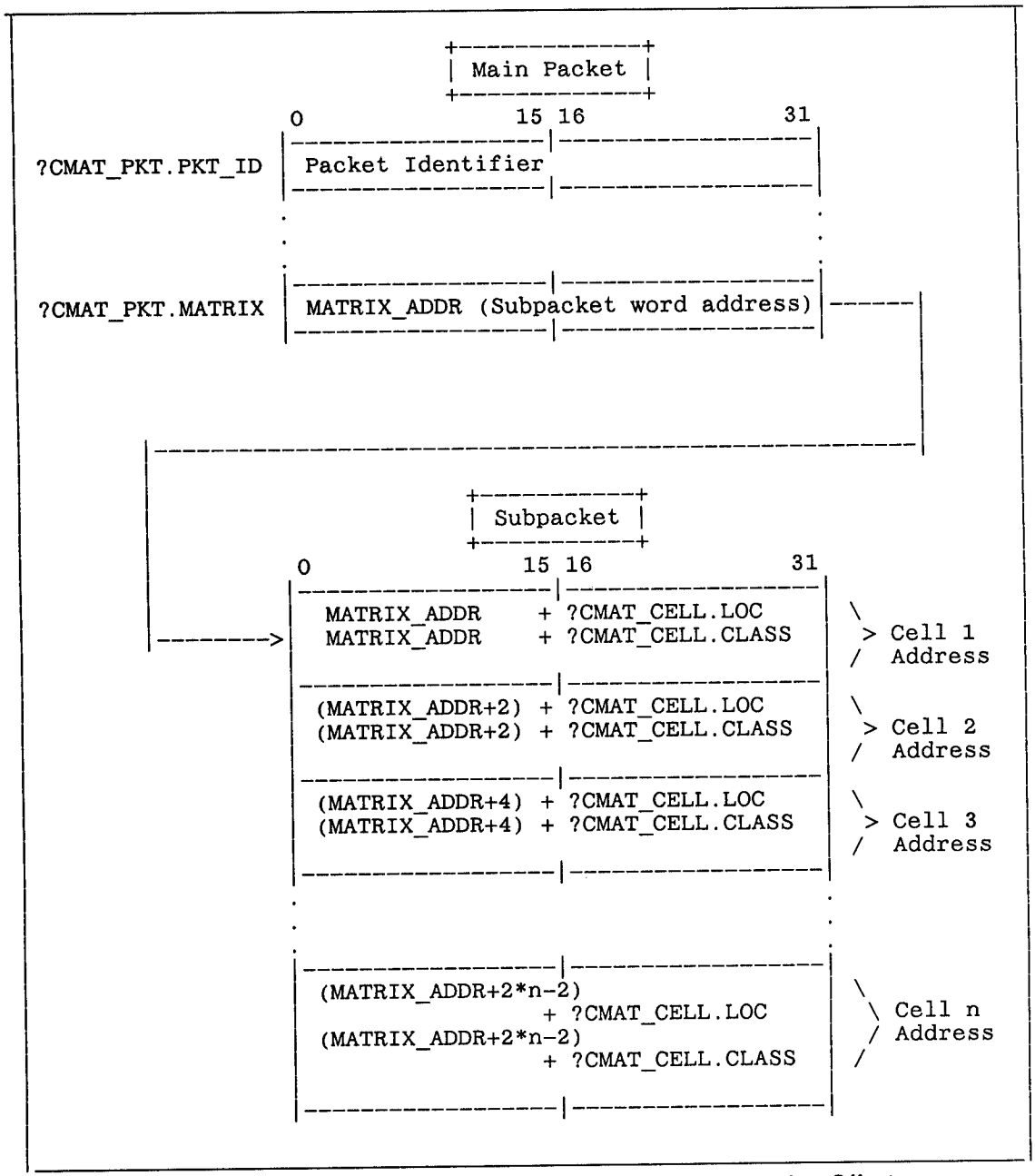


Figure 2–13. Addresses of ?CMATRIX Main Packet and Its Subpacket Offsets

Licensed Material - Property of Data General Corporation

?CON Becomes a customer of a specified server.

?CON
error return
normal return

Input

AC0 One of the following:

- PID of the server
- Byte pointer to the server's process name

AC1 Bit masks:

?MCPID	if AC0 contains a byte pointer
?MCOBIT	(optional) to suppress the obituary message when the server issues ?DCON, ?RESIGN, or ?CHAIN, or terminates
?MCRNG	if AC2 contains a ring number (If ?MCRNG is not set, the O/S assumes Ring 7.)

AC2 Contains the following:

- Bits 0 through 28 are reserved (Set to 0.)
- Bits 29 through 31 contain the server ring number if ?MCRNG is set in AC1

Output

AC0 PID of the server

AC1 Unchanged

AC2 Unchanged

Licensed Material - Property of Data General Corporation

Error Codes in AC0

ERCBK	Connection has been broken
ERCCS	Cannot connect to self (AC0 contains the caller's PID, or a byte pointer to the caller's process name.)
ERNAS	Process is not a server (that is, the target process)
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERRNI	Ring number invalid
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?CON establishes a connection so that you can use the connection-management facility. The connection-management facility allows you to issue the IPC system calls, ?ISEND, ?IREC, and ?IS.R, even though the customer process does not have the ?PVIP privilege.

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 tandem.

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 has already broken the 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.

?CPMAX

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

?CPMAX *[packet address]*
error return
normal return

Input

AC0	One of the following: • Byte pointer to the CPD's pathname • 0 if a packet address is supplied
AC1	One of the following: • CPD's new maximum space (MS) value if not supplying a packet • Unused if supplying a packet
AC2	One of the following: • Reserved (Set to 0 if not supplying a packet.) • Address of the ?CPMAX packet

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Licensed Material - Property of Data General Corporation

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

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.

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 offset ?CPMCN in the ?CPMAX packet. The use of a packet is only necessary if you choose to specify the target CPD's channel number. AC0 must be set to zero to indicate the use of a packet.

If a packet is not being used, then AC1 contains the CPD's new maximum space value. If a packet is being used, offsets ?CPMHS and ?CPMLS must be used to indicate the new MS value and AC1 is unused. The calling process must have Write access to the target CPD.

A CPD's MS value defines the maximum number of disk blocks available to the CPD and all of its subordinate files, except for files that are part of a subordinate logical disk (LD).

You cannot set a CPD's MS value so that its current space value (the number of blocks currently allocated) exceeds the MS value of a superior control point, because ?CPMAX will fail on error ERCPD.

Figure 2-14 shows the structure of the ?CPMAX packet.

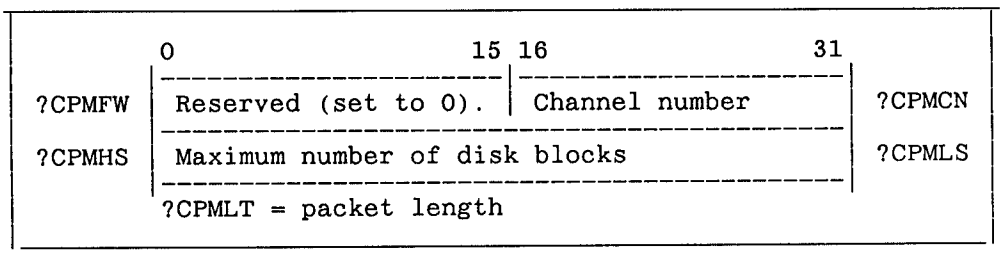


Figure 2-14. Structure of ?CPMAX packet

?CREATE

Creates a file or directory.

?CREATE *[packet address]*
error return
normal return

Input

AC0	Byte pointer to a pathname for the new file
AC1	Reserved (Set to 0.)
AC2	Address of the ?CREATE packet, unless you specify the address as an argument to ?CREATE

Output

AC0	Unchanged
AC1	Undefined
AC2	Address of the ?CREATE packet

Error Codes in AC0 *

ERIFT	Illegal file type
ERIVP	Invalid port number specified
ERLVL	Maximum directory tree depth exceeded
ERMPR	System call parameter address error
ERNAE	Filename already exists
ERVBP	Invalid byte pointer passed as a system call argument
ERWAD	Write access denied
ERWBP	Invalid word pointer passed as a system call argument

Why Use It?

?CREATE allows you to create a file or directory for I/O. (As an alternative, you can create and open a file simultaneously by selecting the creation option in the ?OPEN packet.) ?CREATE is also useful for creating files to serve as ports for IPC messages.

What It Does

You can use ?CREATE to create files of nearly any type, except those that represent certain peripheral devices. (See Table 2–11 for a complete list of the types of files that you can create with ?CREATE.) The ?CREATE caller must have Execute access and either Write or Append access to the target file's directory.

To create a file for the peripheral directory (:PER), the ?CREATE caller must have Write access to the peripheral directory. Otherwise, ?CREATE takes the normal return, but has no effect.

Table 2-11. Valid ?CREATE File Types

Type	Meaning	Comments
?FUDF	User Data File	Usually applies to object files.
?FTXT	Text File	Should contain ASCII text.
?FPRG	AOS Program File	Program file for use under AOS (16-bit code).
?FUNX	MV/UX File	File for use under MV/UX.
?FPRV	AOS/VS Program File	Program file for use under AOS/VS (32- or 16-bit code).
?FDIR	Disk Directory	None.
?FCPD	Control Point Directory	None.
?FLNK	Link File	None.
?FSTF	Symbol Table File	Produced by the Link utility and used primarily by the OS.
?FUPF	User Profile File	Used by PREDITOR (user profile editor) and EXEC.
?FSDF	System Data File	You cannot specify record format types (left byte of offset ?CFTYP) for system data files.
?FIPC	IPC Port Entry	IPC file.
?FGLT	Generic Labeled Tape	None.
?FSPR	Spoolable Peripheral Directory	None.
?FQUE	Queue Entry	None.
?FNCC ?FLCC ?FFCC ?FOCC	FORTRAN Carriage Control	None.
?FPIP	Pipe File	None.

?CREATE Continued

Before you issue ?CREATE, you must set up the ?CREATE packet in your address space. You can load the packet address into AC2 before you issue ?CREATE, or you can cite the packet address as an argument to ?CREATE. You must define a pathname for the file, and load AC0 with a byte pointer to that pathname.

To create a link entry with ?CREATE, specify the link name as the last entry in the pathname. Also, place a byte pointer to the link resolution pathname in offset ?CACP of the packet. When you create a link, the operating system does not resolve the pathname associated with the link. Therefore, if the pathname does not exist or if it contains illegal characters, you will not get an error.

The ?CREATE packet varies, depending on whether you are creating an IPC file, a directory, or one of the other valid file types. Figure 2–15, Figure 2–17, and Figure 2–18 show the structures of the three ?CREATE packets.

IPC Entry Packet

You can create an IPC file only in the ?CREATE caller's initial working directory. When a process terminates, the operating system deletes all of its IPC files. Figure 2–15 shows the ?CREATE packet for creating an IPC file and Table 2–12 describes its contents.

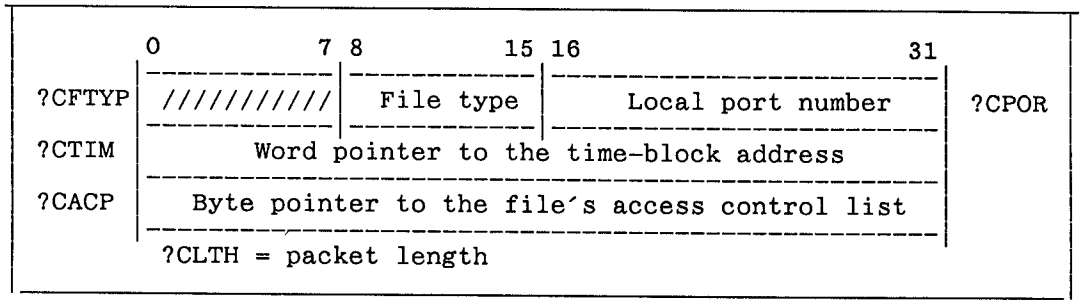


Figure 2–15. Structure of ?CREATE IPC Packet

Set offset ?CFTYP to the file type for the IPC file. The valid file types for IPC files are ?FIPC, which represents a standard IPC file, and ?FSPR, which represents an IPC file for a spoolable peripheral device. You must set this offset to ?FIPC or ?FSPR; there is no default value.

Table 2-12. Contents of ?CREATE IPC Packet*

Offset	Contents
?CFTYP	(Right byte.) File type for IPC files: ?FIPC (standard IPC file) or ?FSPR (IPC file for a spoolable peripheral device).
?CPOR	Local port number for IPC file in range from 1 through ?MXLPN.
?CTIM (double word)	Address of new file's time block, if present. (See Figure 2-16 for the structure of the ?CREATE time block.) DEFAULT = -1 (Set all values to current date and time).
?CACP (double word)	New file's access control list (ACL), which you specify elsewhere in your program. If you set ?CACP to 0, the new file will have no ACL. (See "Access Control Specifications" for more information on ACLs.) DEFAULT = -1 (give new file caller's default ACL).

* There is no default unless otherwise specified.

Use offset ?CPOR to define a local port number for the IPC file. The legal range for local port numbers is 1 through ?MXLPN. Offset ?CPOR has no default value. (The ring from which ?CREATE is issued determines the ring number that is associated with the public port.)

Offset ?CTIM points to the address of the new file's time block, if one exists. The time block is an additional parameter you can include to record the time the file was created and when it was last accessed and/or modified. To set all values within the time block to the current time, set ?CTIM to -1 (the default value).

Figure 2-16 shows the structure of the ?CREATE time block.

You must express the time-block values as double-precision integers where the high-order portion states the duration (in days) from 31 December 1967 to the date the file was created, accessed, or modified. The low-order portion states the time the file was created, accessed, or modified. You express the time value as the number of biseconds (half the number of seconds) since midnight.

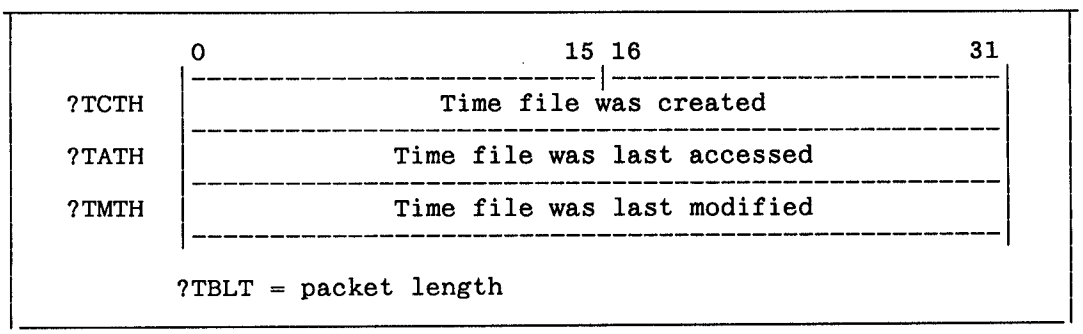


Figure 2-16. Structure of ?CREATE Time Block

Access Control Specifications

Within all three packet types, offset ?CACP points to the new file's access control list (ACL), which you must specify elsewhere in your program. If you set this parameter to -1 (the default value), the new file will have the ?CREATE caller's default ACL. If you set ?CACP to 0, the new file will have no ACL. If you specify an ACL, use the format shown in the description of ?DACL.

Directory Packet

Figure 2-17 shows the ?CREATE packet for creating a directory and Table 2-13 describes its contents. As in the other ?CREATE packets, the right byte of offset ?CFTYP specifies the new directory's file type: either ?FDIR, for a standard directory, or ?FCPD, for a control point directory.

Offsets ?CTIM and ?CACP point to the addresses of the directory's time block and ACL, respectively. The standard default for ?CTIM is -1, which sets all values in the time block to the current time. (See Figure 2-16 for the structure of the time block. Also, see "Access Control Specifications," for a description of ?CACP.)

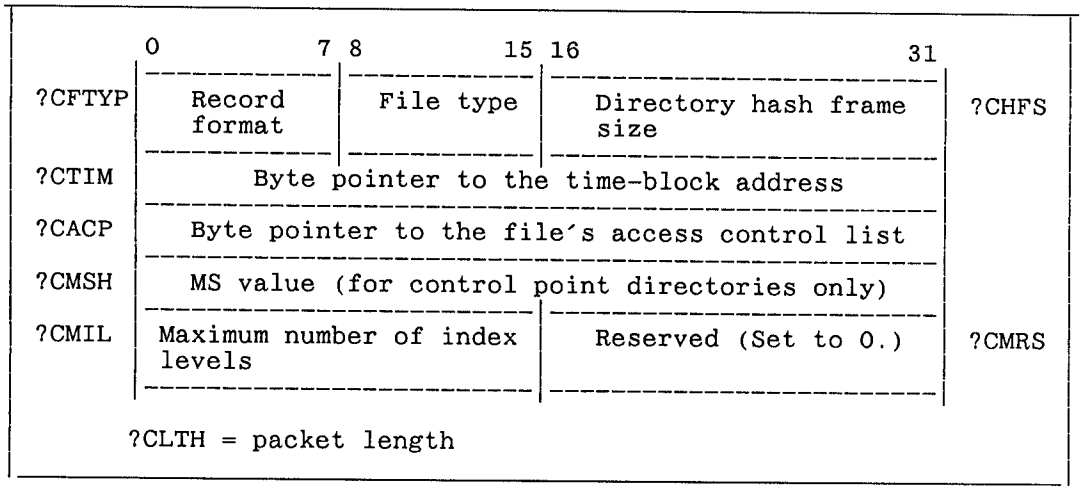


Figure 2-17. Structure of ?CREATE Directory Packet

If you are creating a control point directory (CPD), set offset ?CMSH to the maximum space value (MS) for the directory. The MS value is the maximum number of disk blocks available to the CPD and all its subordinate files, excluding files in a subordinate logical disk. (To change a CPD's MS value, use the ?CPMAX system call.)

The operating system always creates a value of 3 for the maximum number of index levels in any new directory file that it creates. The default value of -1 for ?CMIL, like all values, results in this maximum number.

If you set offset ?CMIL to 0 (no index levels), the operating system builds the directory contiguously. If you set ?CMIL to -1 (the default value) or to a value outside the legal range, the operating system creates as many index levels as necessary, to a maximum of three levels.

If you set offset ?CHFS to -1 or 0, the operating system assigns the default length hash frame size to the directory.

Table 2-13. Contents of ?CREATE Directory Packet*

Offset	Contents
?CFTYP	(Right byte.) File type of new directory: ?FDIR (for a standard directory) or ?FCPD (for a control point directory).
?CHFS	Directory hash frame size. DEFAULT = -1 (Set hash frame size to 7.)
?CTIM (double word)	Address of new file's time block, if present. (See Figure 2-16 for the structure of the ?CREATE time block.) DEFAULT = -1 (Set all values to current date and time).
?CACP (double word)	New directory's access control list (ACL), which you specify elsewhere in your program. If you set ?CACP to 0, the file will have no ACL. DEFAULT = -1 (give new file caller's default ACL).
?CMSH (double word)	(Control point directories only. Set to 0 for file type ?FDIR.) Maximum space value of new directory.
?CMIL	Maximum number of index levels for new directory. The operating system always creates 3. DEFAULT = -1 (set maximum number of index levels to 3).
?CMRS	Reserved (set to 0).

* There is no default unless otherwise specified.

?CREATE Continued

Other File Types Packet

Figure 2–18 shows the structure of the packet for the remaining file types, and Table 2–14 describes its contents.

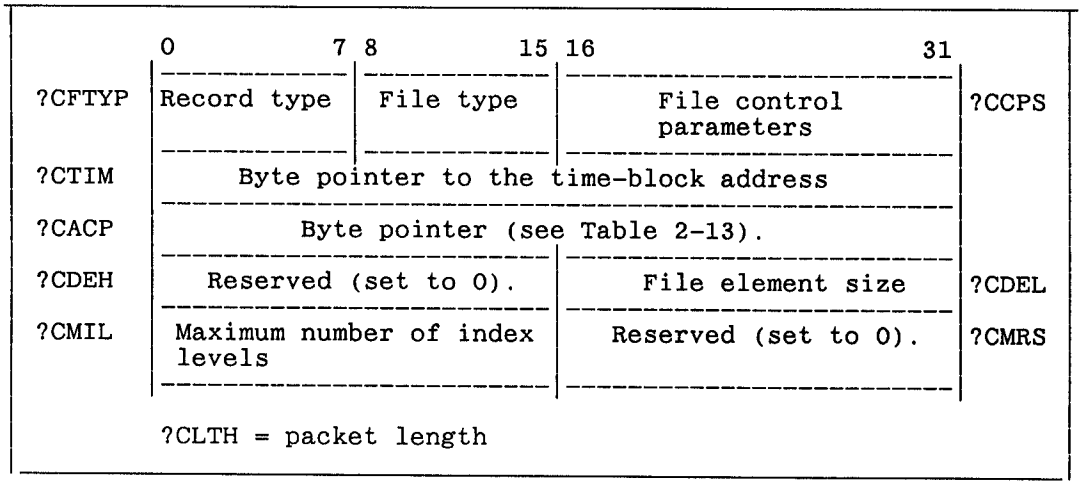


Figure 2–18. Structure of ?CREATE Packet for Other File Types

In this packet, offset ?CFTYP contains two significant parameters: record format and file type. You can specify any valid file type, except the following:

- ?FIPC or ?FSPR (use the ?CREATE IPC packet).
- ?FDIR or ?FCPD (use the ?CREATE directory packet).
- Any file type that is restricted from user creation.

You can set the record format field to any one of the following masks:

?ORDY	dynamic
?ORDS	data-sensitive
?ORFX	fixed-length
?ORV	variable-length

Table 2-14. Contents of ?CREATE Packet for Other File Types*

Offset	Contents
?CFTYP	(Left byte: record type.) This can be any one of the following masks: ?ORDY (dynamic); ?ORDS (data-sensitive); ?ORFX (fixed-length); or ?ORVR (variable-length). (Right byte: file type.) This can be any file type except ?FIPC, ?FSPR, ?FDIR, or ?FCPD.
?CCPS	(Fixed-length records only; otherwise ignored.) Maximum record length of new file.
?CTIM (double word)	Address of new file's time block, if present. (See Figure 2-16 for the structure of the ?CREATE time block.) DEFAULT = -1 (set all values to current date and time).
?CACP (double word)	One of the following: - Byte pointer to new file's access control list (ACL). (If you set ?CACP to 0, the new file will have no ACL.) - Byte pointer to link resolution pathname if you are creating a link. DEFAULT = -1 (give new file caller's default ACL).
?CDEH	Reserved (set to 0).
?CDEL	Number of disk blocks per element for new file in the range from -1 through 65534. DEFAULT = -1 (set to element size selected during the system-generation procedure).
?CMIL	Maximum number of index levels for new directory in the range from 0 through 3. DEFAULT = -1 (set maximum number of index levels to 3).

* There is no default unless otherwise specified.

If you do not specify the record format, you must specify it when you open, read, or write the file.

Offset ?CCPS, the file control parameter, applies only to entries with fixed-length records. Offset ?CCPS must equal the new file's maximum record length. The operating system ignores ?CCPS if the record format is not fixed length.

Offset ?CTIM points to the address of the directory's time block. The standard default for ?CTIM is -1, which sets all values in the time block to the current time. (See Figure 2-18 for the structure of the time block.)

Offset ?CAP either points to the directory's ACL or it points to the resolution pathname of a link. (See "Access Control Specifications," in this description for more information on using offset ?CACP as a byte pointer to the ACL.)

Set offset ?CDEL to the new file's file-element size; that is, the number of disk blocks per file element. If you set this parameter to -1, the operating system assigns the default file-element size. (The default file-element size was specified during the system-generation procedure.)

The operating system rounds the file-element size you specify to the next multiple of the default file-element size. If you specify a file-element size larger than 65534, the operating system assigns the file a file-element size of 65534. If you default ?CDEL, the operating system assigns the default file-element size to the entry.

If you set ?CMIL to 0 (no index levels), the operating system builds the directory contiguously. If you set ?CMIL to -1 (the default value) or to a value outside the legal range, the operating system creates as many index levels as necessary, to a maximum of three levels.

You can use ?CREATE to create a pipe file whose default size is 4096 bytes (2 pages), in which case offsets ?CFTYP, ?CCPS, and ?CDEL are ignored. However, ?OPEN offers you control over a pipe file's length (specify up to 20 pages in offset ?IMRS) and pending behavior. Only ?OPEN lets you use the pipe extension packet. Details appear under ?OPEN.

Sample Packet

The following sample packet shows creating an IPC file:

```
PKT:    .BLK?       CLTH                ;Allocate enough space for packet
                                        ;(packet length = ?CLTH).

        .LOC        PKT+?CFTYP          ;File type for IPC file.
        .WORD       ?FIPC               ;Standard IPC file. (There is no
                                        ;default for ?CFTYP.)

        .LOC        PKT+?CPOR           ;Local port number.
        .WORD       5                   ;Local port number is 5.

        .LOC        PKT+?CTIM           ;Address of time block.
        .DWORD      -1                  ;No time block exists, so set all
                                        ;values in time block to current time
                                        ;(default = -1).

        .LOC        PKT+?CACP           ;Byte pointer to file's ACL.
        .DWORD      -1                  ;Give the new file the same ACL as
                                        ;the caller.

        .LOC        PKT+?CLTH           ;End of ?CREATE IPC packet.
```

Notes

- See the description of ?OPEN in this chapter for information on the creation option.
- See the descriptions of ?OPEN, ?READ, and ?WRITE in this chapter for information on the structure of their packets.

?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 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 ?CRUDA packet, unless you specify the address as an argument to ?CRUDA

Output

AC0 Unchanged

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

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.

What It Does

A User Data Area (UDA) is often used to store a file's format descriptions, although you can use it for any other purpose. The ?CRUDA system call creates a 128-word UDA for the target file that you specify. The target file can be specified in one of two ways: either by a byte pointer to the file's pathname in AC0, or by using offset ?GCPCN in the ?CRUDA packet. The use of the packet is only necessary if you choose to specify the target file's channel number. AC0 must be set to zero to indicate the use of a packet.

The calling process must have Write or Owner access to the target file. Once you create a UDA with ?CRUDA, you can write to it with ?WRUDA and read from it with ?RDUDA. The UDA exists until you delete the target file.

Figure 2-19 shows the structure of the ?CRUDA packet.

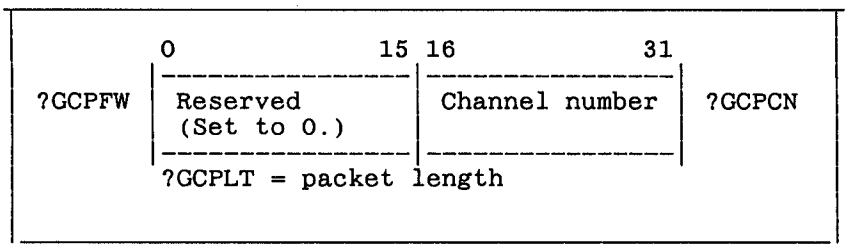


Figure 2-19. Structure of ?CRUDA packet

Notes

- See the descriptions of ?RDUDA and ?WRUDA in this chapter.

?CTERM — Terminates a customer process.

?CTERM
error return
normal return

Input

AC0	PID of the customer to terminate or a byte pointer to that customer's process name	Unchanged
AC1	If AC0 contains a byte pointer, -1	Unchanged
AC2	Reserved (Set to 0.)	Unchanged

Output

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection doesn't exist
ERPRH	Attempt to access process not in the hierarchy
ERPNM	Illegal process name
ERPRV	Caller not privileged for this action (The caller is not a server of the designated customer.)
ERVBP	Invalid byte pointer passed a system call argument

Why Use It?

?CTERM is one of six disconnect system calls. (The others are ?BRKFL, ?DCON, ?RESIGN, ?RETURN, and ?TERM.) Although ?CTERM breaks the customer/server connection, it does not clear the customer/server entry from the connection table. Both processes must disconnect for this to occur.

What It Does

?CTERM terminates the customer process that you specify in AC0, and breaks all its connections with servers. The calling process must be a server of the customer that you specify.

When a server issues ?CTERM, the operating system returns an obituary message from the global IPC port ?SPTM. The obituary message notifies the server of the customer's demise.

To receive the obituary message, the server must issue a global ?IREC against port ?SPTM. The operating system passes the obituary message (termination code ?TBCX) to offset ?IUFL in the server's ?IREC header. (If the server is the customer's father, it must issue two ?IREC system calls: one for the obituary message, and one for the termination code.)

Notes

- See the description of ?IREC in this chapter.
- See the descriptions of ?BRKFL, ?CON, ?DCON, ?RESIGN, ?RETURN, and ?TERM (the disconnect system calls) in this chapter.

?CTOD

Converts a scalar time value.

?CTOD
error return
normal return

Input

AC0	Scalar time you want to convert expressed as the number of biseconds (seconds/2) since midnight in Bits 16 through 31
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Seconds from 0 through 59
AC1	Minutes from 0 through 59
AC2	Hour from 0 (midnight) through 23 (11 p.m.). (The result is expressed in octal.)

Error Codes in AC0

No error codes are currently defined.

Why Use It?

The system clock maintains the current time in the standard hours, minutes, and seconds, where the value for the current hour can range from 0 (midnight) through 23 (11 p.m.). Both the ?FSTAT packet and the ?CREATE time block express the time in scalar notation. Thus, you can use ?CTOD to convert the values returned by ?FSTAT or ?CREATE.

What It Does

?CTOD converts a scalar time value, specified in AC0, to its equivalent in hours, minutes, and seconds. The scalar value for a particular time is the number of biseconds (half the number of seconds) that have elapsed since midnight.

?CTYPE

Changes a process type.

?CTYPE
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the name of the target process
- PID of the target process
- −1 to change the process type of the calling process

AC1 One of the following:

- −1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID

AC1 is ignored if AC0 contains −1

AC2 New process type. The process type parameters are:

- 0 for swappable processes
- ?PFRP for pre-emptible processes
- ?PFRS for resident processes

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Licensed Material - Property of Data General Corporation

?CTYPE Continued

Error Codes in AC0

ERMPR	System call parameter address error
ERPNM	Illegal process name format
ERPRH	Attempt to access a process not in hierarchy
ERPRP	Illegal process priority
ERPTY	Illegal process type
ERVBP	Invalid byte pointer passed as a system call argument
ERWSF	Working set not swappable (The working set has too many wired pages and, therefore, this process cannot become nonresident.)

Why Use It?

?CTYPE lets you change a process's type during program execution, thereby possibly changing the process's priority for a system resource. You can use ?CTYPE to favor the target process or to favor another process over the target process.

What It Does

?CTYPE changes the process type of the process that you specify in AC0 (either the calling process itself or one of its sons). To issue ?CTYPE, the calling process must have privilege ?PVTY assigned in the ?PROC packet. If the calling process is in Superprocess mode, it can use ?CTYPE to change the type of any process, not just those in its own process tree.

NOTE: When a 16-bit process becomes resident, the operating system implicitly wires its ring 7 pages, and then unwires them if the process changes to another process type.

Notes

- See the descriptions of ?PROC and ?WIRE in this chapter.

?DACL

Sets, clears, or examines a default access control list.

?DACL
error return
normal return

Input

AC0 One of the following:

- −1 to set a new default ACL to the string that the byte pointer in AC1 specifies
- 0 to return the current default ACL string, and the status (DEFACL ON or OFF) in AC0
- 1 to turn off the default ACL mode

AC1 Byte pointer to a 128-word buffer if AC0 contains −1; if AC0 = 0, then AC1 contains a byte pointer to a 128-word receive buffer

AC2 Reserved (Set to 0.)

Output

AC0 The default ACL mode (on or off), if AC0 = 0 on input; otherwise, unchanged

AC1 Unchanged

AC2 Undefined

Error Codes in AC0

ERACL Illegal ACL
ERPRE Invalid system call parameter
ERVBP Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?DACL to assign your process and one or more other processes (which you specify by their usernames) access to all files that your process will create. A default ACL that you define this way is valid until the ?DACL caller terminates or issues another ?DACL to redefine the default ACL.

Licensed Material - Property of Data General Corporation

?DACL Continued

What It Does

Depending on the input value of AC0, ?DACL lets you define a default ACL for the duration of the calling process, clear the default ACL, or obtain the current default ACL. To supply an ACL string, use the following format:

username<0>accesstype[...]<0>

where

accesstype is one of the following:

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

Two examples of supplied ACL strings are

JEFF<0><?FACO+?FACW+?FACA+?FACR+?FACE><0>

and

JEFF<0><?FACO+?FACW+?FACA+?FACR+?FACE>LISA<0><?FACR+?FACE><0>

If you do not supply an ACL string, the operating system sets the default ACL to full access for the calling process and all processes that have the calling process's username:

caller's_username<0><?FACO+?FACW+?FACA+?FACR+?FACE><0>

To examine the current ACL, load AC0 with 0. The operating system then returns the current default ACL to the buffer you specify in AC1.

Note that the default ACL keys on specific usernames, that is, any usernames that the calling process cites in the ACL string, rather than keying on specific filenames.

?DADID — Gets the PID of a process's father.

?DADID
error return
normal return

Input

AC0	One of the following: • PID of the target process • −1 to obtain the PID of the caller's father process
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	PID of the target process's father
AC2	Undefined

Error Codes in AC0

ERPRH	Attempt to access process not in hierarchy
ERPOR	PID is out of range for this process

Why Use It?

A number of system calls require a PID as an input parameter. ?DADID provides you with a simple way to identify a PID, which you can then use as an input parameter for another system call.

What It Does

?DADID returns the PID of a target process's father. The target process, which you specify in AC0, can be either the caller or some other process in the process tree.

?DCON Breaks a connection (disconnects) in Ring 7.

?DCON
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	PID of the target process in the connection
AC2	Reserved (Set to 0.)

Output

AC0	PID of the server
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERCBK	Connection has been broken
ERCDE	Connection doesn't exist (Two likely errors: the target process does not exist; or the target process is not connected with the caller.)

Why Use It?

?DCON breaks the Ring 7 connection from either the caller's or the server's end. (To break a connection to a specific ring other than Ring 7, use ?DRCON.)

What It Does

?DCON breaks the Ring 7 connection between the calling process and the process (customer or server) that you specify in AC1. Both servers and customers can issue ?DCON.

Note that a single ?DCON does not clear the customer/server connection from the connection table. For this to occur, both processes must break the connection. If both processes do not break the connection, the operating system reserves the PID number and does not reuse it until the connection is cleared.

Notes

- See the description of ?CON (the connection system call) and the descriptions of ?BRKFL, ?CTERM, ?DRCON, ?RESIGN, and ?RETURN (the disconnect system calls) in this chapter.
- ?DCON is provided primarily to provide compatibility with 16-bit programs converted from AOS. ?DRCON is the preferred system call.

?DDIS

Disables access to all devices.

?DDIS
error return
normal return

Input

None

Output

None

Error Codes in AC0

ERPRV Caller not privileged for this action

Why Use It?

?DDIS is the opposite of ?DEBL, which enables access to all user-defined and system-defined devices. You can issue ?DDIS to disable device access after completing a task I/O routine that began with ?DEBL.

What It Does

?DDIS disables a process's access to all system and user devices.

The calling process must have privilege ?PVDV to issue ?DDIS successfully. Also, ?DDIS does not re-enable LEF mode. Therefore, to re-enable LEF mode, you must issue ?LEFE (enable LEF mode).

To re-enable device I/O after a ?DDIS, issue ?DEBL.

?DDIS affects only the segment that issued the call.

?DDIS turns off the "issue I/O instructions" privilege for a segment. See the *MV/Family Principles of Operations* manual.

?DEASSIGN — Cancels a character device.

```
?DEASSIGN
error return
normal return
```

Input

AC0	Byte pointer to the name of the device that you want to deassign
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERARC	Attempt to release console device
ERARU	Attempt to release an unassigned device
ERIDT	Illegal device name type
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

If you used ?ASSIGN to assign a character device to a process, you must use ?DEASSIGN to cancel the assignment. The operating system also allows implicit device assignments and cancellations with ?OPEN and ?CLOSE.

What It Does

?DEASSIGN breaks the explicit assignment you established between the calling process and a device with ?ASSIGN. Note that you cannot use ?DEASSIGN to call the process's console or any character device that is currently open.

Notes

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

?DEBL

Enables access to all devices.

?DEBL
error return
normal return

Input

None

Output

None

Error Codes in AC0

ERPRV Caller not privileged for this action

Why Use It?

You would probably issue ?DEBL before entering a task-level I/O sequence. ?DEBL gives the calling process global access to all devices, including system-defined devices.

What It Does

?DEBL turns off I/O protection and disables LEF mode. This allows a program to issue instructions to *any* I/O device without operating system intervention.

Because ?DEBL turns off I/O protection and disables LEF mode, tasks within the calling process can successfully issue I/O instructions against devices. However, the calling process must have the ?PVDV privilege ("access devices") to issue ?DEBL.

After the operating system executes ?DEBL, it interprets all subsequent LEF instructions as I/O instructions, until you issue ?LEFE to restore LEF mode.

?DEBL affects only the segment that issued the call.

?DEBUG

Calls the Debugger utility.

?DEBUG
error return
normal return

Input

None

Output

None

Error Codes in AC0

No error codes are currently defined.

Why Use It?

If you pass control to the user debugger directly from a program, you can examine memory locations and/or execution paths while the program runs. (Another way to call the Debugger utility is to choose the ?PFDB option in offset ?PFLG of the ?PROC packet.)

What It Does

?DEBUG passes control to the operating system's Debugger utility while your program is executing. To return control to the program (that is, to the ?DEBUG normal return), issue the Debugger command $P (ESC P).

Notes

- See the description of ?PROC in this chapter.

Licensed Material - Property of Data General Corporation

?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 O/S 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.

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
ERCDR	Can't delete the root directory

Licensed Material - Property of Data General Corporation

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.

What It Does

?DELETE deletes a file or directory. The target entry can be specified in one of two ways: either by a byte pointer to the entry's pathname in AC0, or by using offset ?GCPCN in the ?DELETE packet. The use of a packet is only necessary if you choose to specify the target entry's channel number. AC0 must be set to zero to indicate the use of a packet.

The calling process must have Write access to the target entry's directory or Owner access to the target entry.

If a byte pointer to the entry's pathname is specified in AC0 and if the last file in the pathname is a link entry, then the operating system will delete the link entry without resolving it. Therefore, the directories or files to which the link refers remain intact.

As the error codes indicate, you cannot delete a process's working directory or any directory cited in a search list.

Figure 2–20 shows the structure of the ?DELETE packet.

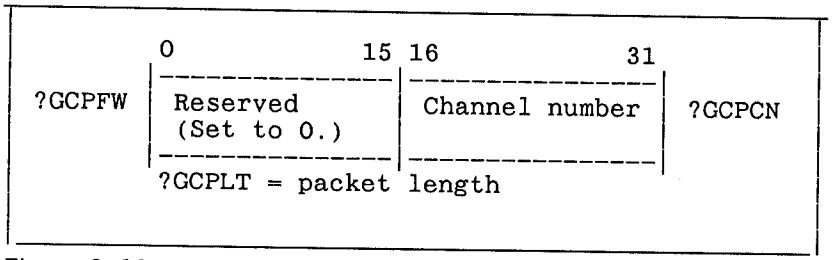


Figure 2–20. Structure of ?DELETE Packet

Notes

- See the description of ?CREATE in this chapter for information on creating files and directories.

?DEREGISTER

Deletes an entry from the Public Name Registry.

AOS/DVS only

?DEREGISTER
error return
normal return

Input

AC0	Byte pointer to the public name you are deleting
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERODE	Object does not exist
EROIU	Object in use
ERONR	Object not registered

File system errors

Why Use It?

Use this system call to delete an entry for the specified resource from the Public Name Registry.

For example, a server that wishes to respond to requests from customers in a distributed environment could identify itself by using ?REGISTER. This makes its well-known IPC port name globally visible (in the community). Potential customers could then issue ?ILKUP calls against this well-known public name. Such a server would delete (?DEREGISTER) the port name as part of its normal termination processing.

Licensed Material - Property of Data General Corporation

?DEREGISTER Continued

What It Does

A resource becomes globally visible to users on all nodes in an AOS/DVS community as a result of its (or a containing directory's) being identified to the Public Name Registry (PNR) and the PNR's creating an appropriate entry in its Public Name Registry. The ?REGISTER system call lets you identify resources that will become globally visible. In contrast, the ?DEREGISTER system call lets you remove a currently registered resource from the PNR's registry.

AOS/DVS treats multiple attempts to cancel (?DEREGISTER call) two different resources under the same globally visible name as an error.

?DFRSCH — Disables task rescheduling and indicates prior state of rescheduling.

?DFRSCH
error return
normal return

Input

None

Output

AC0 ?DSCH if task rescheduling is disabled when you issue ?DFRSCH

AC1 Undefined

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?DFRSCH to lock all other tasks out of a critical region and still retain a copy of the previous state of multitask scheduling in memory. Internally called subroutines that need to disable scheduling can use ?DFRSCH, rather than ?DRSCH, to restore previous multitask scheduling environments when the operating system exits from the subroutine.

As with ?DRSCH, you should be very careful when you use ?DFRSCH, because ?DFRSCH disrupts multitask activity for the entire process.

What It Does

?DFRSCH blocks all multitask scheduling for the current process until you explicitly re-enable scheduling with ?ERSCH or until you terminate the calling task with ?KILL. The calling task retains CPU control while scheduling is disabled, even if other tasks are ready to run.

When ?DFRSCH succeeds, it returns a flag in AC0 that indicates whether multitask scheduling was enabled before ?DFRSCH took action. A value of ?DSCH in AC0 indicates that multitask scheduling was disabled before you issued ?DFRSCH.

Notes

- See the descriptions of ?ERSCH and ?DRSCH in this chapter.

?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 directory
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERDAD	Directory access denied
ERVBP	Invalid byte pointer passed as a system call argument

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 allows you to return to your initial directory after you work elsewhere.

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). The calling process must have Execute access to a directory to use it as a working directory.

?DISSOC — Disassociates two processes.

?DISSOC *[packet address]*
error return
normal return

See the description of ?ASSOC in this chapter for further information on ?DISSOC.

?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 allows you to 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.

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 2-21.) 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.

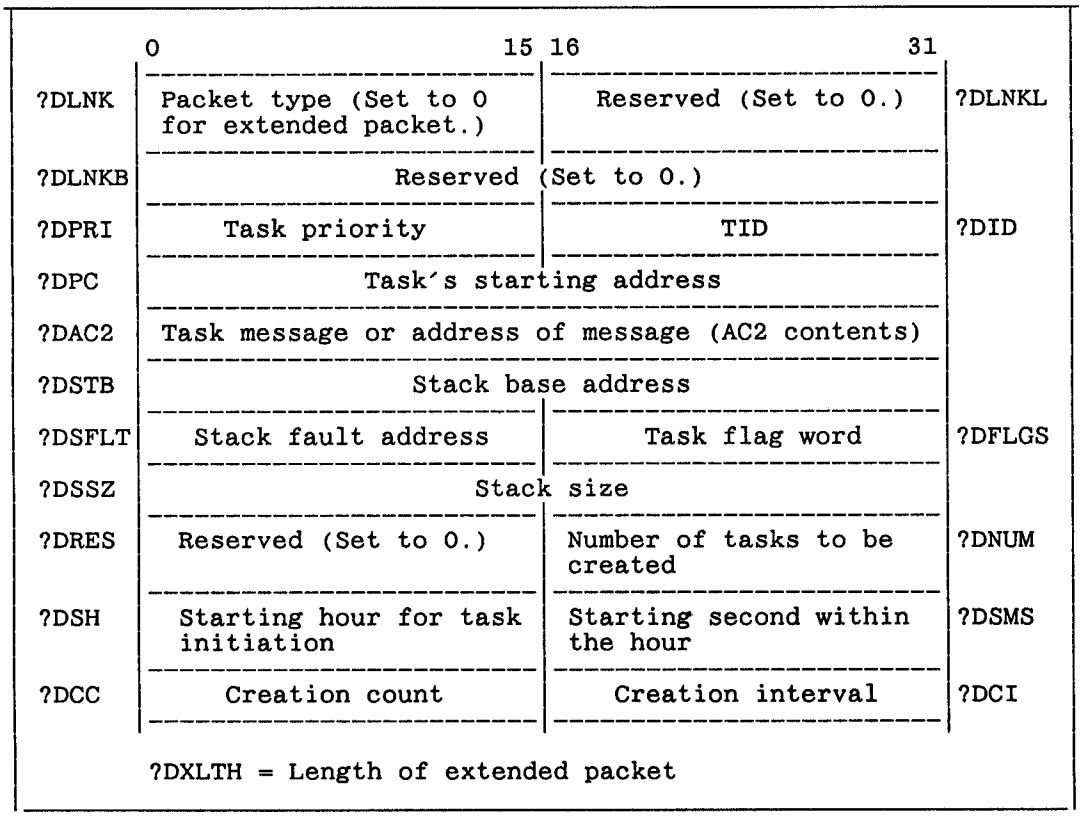


Figure 2-21. Extended Task Definition Packet

If you issue ?DQTSK against an active task, it remains active. The operating system simply pulls the task definition packet from the queue, if it is on the queue. If you want to requeue the task, you must issue ?TASK again.

Notes

- See the descriptions of ?IQTSK and ?TASK in this chapter.

?DRCON Breaks a connection (disconnects).

?DRCON
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	PID of the target process in the connection
AC2	Contains the following: • Bits 0 through 28 are reserved (Set to 0.) • Bits 29 through 31 contain the ring of the target in the specified process

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERCBK	Connection has been broken
ERCDE	Connection doesn't exist (Two likely errors: the target process does not exist; or the target process is not connected with the caller.)
ERRNI	Invalid ring number

Why Use It?

You can use ?DRCON to break a connection to a specific ring from either the customer's or the server's end.

What It Does

?DRCON is similar to ?DCON in that it breaks the connection between the calling process and the process (customer or server) that you specify in AC1. However, unlike ?DCON, ?DRCON allows you to specify the ring field in AC2. Both servers and customers can issue ?DRCON.

Note that a single ?DRCON does not clear the customer/server connection from the connection table. For this to occur, both processes must break the connection. If both processes do not break the connection, the operating system reserves the PID number and does not re-use it until the connection is cleared.

Notes

- See the description of ?CON and ?DCON in this chapter.

?DRSCH — Disables scheduling.

?DRSCH
error return
normal return

Input

None

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?DRSCH to lock all tasks, except the calling task, out of a critical region. In contrast, ?XMT, ?REC, ?IDSUS, ?PRSUS, ?IDRDY, and ?PRRDY give you finer control over resource allocation. Use ?DRSCH with discretion, if at all; ?DRSCH disrupts multitasking activity for the entire process.

What It Does

?DRSCH suppresses all multitask scheduling for the current process until you explicitly re-enable scheduling with ?ERSCH or terminate the calling task with ?KILL. The calling task retains CPU control while scheduling is disabled, even if other tasks are ready to run. (Note that if the calling task becomes suspended, so will the entire process.)

Notes

- See the descriptions of ?DFRSCH, ?ERSCH, and ?KILL in this chapter.

?ENBRK

Enables a break file.

?ENBRK
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the name of the target process
- PID of the target process
- −1 to enable a break file for the calling process

AC1 One of the following:

- −1 if AC0 is a byte pointer
- 0 if AC0 is a PID
- Ignored if AC0 contains −1

AC2 One of the following:

- Address of the ?ENBRK packet
- −1 to cause a default dump of Ring 7 unshared and shared

Output

AC0 Unchanged or error code

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERNBK	No break file enabled for this ring
ERPRE	Invalid argument passed as system call parameter
ERPRV	Caller not privileged for this action
ERRNI	Invalid ring
ERVBP	Invalid byte pointer passed as system call parameter
ERVWP	Invalid address passed as system call parameter

Licensed Material - Property of Data General Corporation

Why Use It?

?ENBRK allows you to get a memory image of any user ring of a process if the process traps, issues a CTRL-C CTRL-E, or is the target of a TERM/BREAK. This memory image is useful in debugging programs.

What It Does

?ENBRK enables the operating system to take a break file for the specified ring if the target of the ?ENBRK traps, does a CTRL-C CTRL-E, or is the target of a TERM/BREAK. ?ENBRK allows you to specify a ring to dump, a filename to dump to, or a directory in which to create a dump with a default filename.

You can issue ?ENBRK against the calling process for all rings greater than or equal to the ring of the task that is making the call. No other privileges are necessary. Also, you can issue ?ENBRK against another process if the calling process has Superprocess set on or is a server with a valid connection to the target.

When a server issues ?ENBRK against a customer, the ring that you specify as the one you want to dump must be greater than or equal to the ring of the connection. Note that the hierarchy structure is not sufficient to issue ?ENBRK. This means that a process cannot issue ?ENBRK against its son unless it has Superprocess set on, or unless it has a valid connection to its son.

An ?ENBRK overrides all previous ?ENBRK system calls to the specified ring of the process.

The target process must have Write access to the directory for which the break file is intended at the time the process terminates. The target process must also have Read access to the ring .PR programs that will be dumped. If the target process is denied directory access or if the CPD maximum is exceeded, then the operating system does not create the break file.

?ENBRK Continued

To use ?ENBRK, you must supply a packet. Figure 2-22 shows the structure of the packet and Table 2-15 describes its contents.

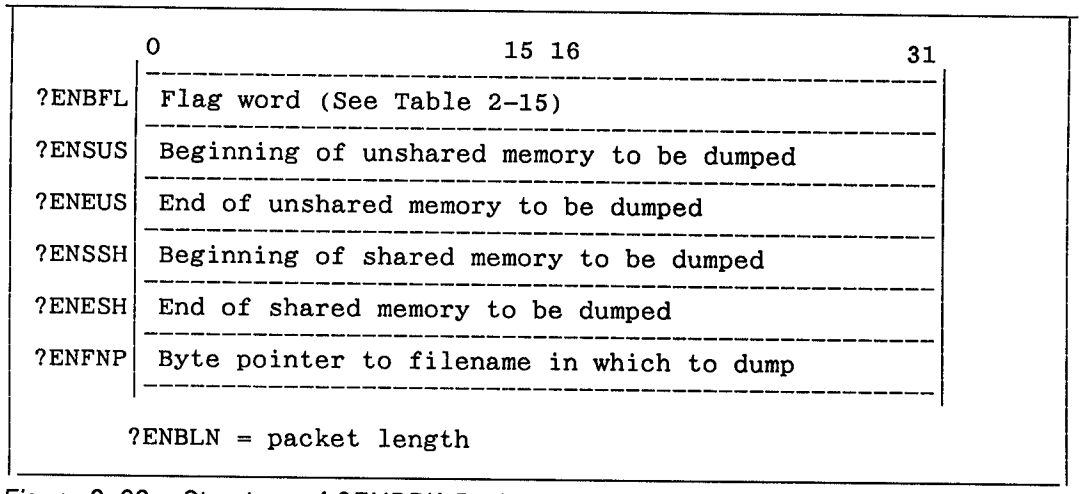


Figure 2-22. Structure of ?ENBRK Packet

Table 2-15. Contents of ?ENBRK Packet*

Offset	Contents
?ENBFL (double word)	Flag word ?ENUS---Dump unshared memory. ?ENSH---Dump shared memory. ?ENPRE--Dump preamble information. (You will get a preamble if you specify either ?ENUS or ?ENSH.) ?ENCST--Return current status. ?ENDIR--Indicate that ?ENFNP is a byte pointer to a directory. (If you do not set this flag, ?ENFNP must be a byte pointer to a pathname.) ?ENR4---Enable break file for Ring 4. ?ENR5---Enable break file for Ring 5. ?ENR6---Enable break file for Ring 6. ?ENR7---Enable break file for Ring 7.
?ENSUS (double word)	Beginning of unshared memory to be dumped. DEFAULT= -1 (to start dumping at beginning of unshared area).

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

Table 2-15. Contents of ?ENBRK Packet*

Offset	Contents
?ENEUS (double word)	End of unshared memory to be dumped. DEFAULT= -1 (to dump to end of unshared area).
?ENSSH (double word)	Beginning of unshared memory to be dumped. DEFAULT= -1 (to start dumping at beginning of shared area).
?ENESH (double word)	End of shared memory to be dumped. DEFAULT= -1 (to dump to end of shared area).
?ENFNP (double word)	Byte pointer to filename to dump to. If ?ENDIR is set, ?ENFNP contains a byte pointer to the directory in which you want to create the default dump file. DEFAULT= -1 (to create default dump file in working directory). NOTE: A default filename takes the form ?PID.TIME.RING.BRK where: - PID is the 5-digit PID of the being dumped process. - TIME is the current time in the form hh_mm_ss. - RING is the ring number of the ring being dumped.

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

?ENQUE — Queues a file entry.

?ENQUE
error return
normal return

Input

AC0	Byte pointer to the name of the output queue
AC1	Byte pointer to the specifications string
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERIDT	Illegal device name type
ERPRE	Invalid system call parameter
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?ENQUE allows you to queue files for spooled output even if your system is not running the EXEC process.

What It Does

?ENQUE uses the IPC facility to queue a file entry to the output queue that you specify in AC0. For example, you can use ?ENQUE to queue an entry to a spooler output queue on a system that does not have the EXEC process.

Before you issue ?ENQUE, define a specifications string for the queue request and load AC1 with a byte pointer to this string. The specifications string should contain the pathname of the file to be queued, and can contain specification switches that modify the way it is to be output.

The format for the specifications string is

pathname*[/switch]*<0>

As the format indicates, you must terminate the specifications string with a null. The string can consist of as many as 512 characters, including the terminating null. To pass any switch argument literally, such as a space, which ordinarily terminates the switch, set the high-order bit in this character. Table 2-16 lists the specifications switches, their meanings, and the default specifications (that is, if you omit the switches).

Licensed Material - Property of Data General Corporation

Table 2–16. Switches for ?ENQUE Specifications String

Switch	Meaning	Default Specification
/B	Output is in binary mode.	Output is in text mode.
/D	Delete file after output.	Do not delete file.
/H	Output header at the top of each page.	Do not output header.
/MES=message	Send a message to the operator's console.	Do not send a message.
/P	Pause for the operator's response before starting the file output.	Do not pause.

If you use the /P switch and you omit the /MES switch, the operating system displays the following message on the operator's console before it starts the output:

PAUSE FROM spoolable_devicename

Conversely, if you omit the /P switch and use the /MES switch to pass a message, the operating system displays the following message on the operator's console, followed by your message, before it starts the output:

FROM spoolable_devicename

If you use both the /P and the /MES switches, the operating system displays the following message on the operator's console before it starts the output:

PAUSE FROM spoolable_devicename:message

For example, a typical message might be

PAUSE FROM LPA:TWO-PART FORMS

To start file output, the operator must respond with an appropriate control command, such as

CONTROL @SPOOLER CONTINUE

For more information on these responses, refer to your operating system's "How to Generate and Run" manual.

Notes

- Refer to your operating system's "How to Generate and Run" manual for information on using the spooler.

?ERMSG

Reads the error message file.

?ERMSG
error return
normal return

Input

AC0	Error code associated with the error message text string
AC1 file	Contains the following: • Bits 16 through 23 contain the byte length of the message buffer • Bits 24 through 31 contain the channel number assigned to the error message file (Use 377 for the system default ERMES file.)
AC2	Byte pointer to the message buffer

Output

AC0	Byte length of the error message
AC1	Channel number of the error message
AC2	Unchanged

Error Codes in AC0

ERTXT	Message text longer than specified (Your message buffer is too small.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?ERMSG to get the text string associated with a particular error code.

What It Does

?ERMSG returns the text string associated with an error code (a 32-bit unsigned value) in either the system error message file, ERMES, or in your own error message file.

Licensed Material - Property of Data General Corporation

Before you issue ?ERMSG, perform the following steps:

1. Load AC0 with the error code associated with the message you want to read.

2. Load Bits 16 through 23 of AC1 with the byte size of the buffer you have set aside in your logical address space.

3. Load Bits 24 through 31 with the channel number of the error message file. (If you are reading the default ERMES file, use 377 as the channel number; the operating system returns the actual channel number. If you are reading another error message file, you must open the file with ?OPEN first. Then, you can use the channel number returned in the ?OPEN packet for ?ERMSG.)

If you pass an unknown error code to AC0, ?ERMSG takes the normal return, but the operating system passes the message "UNKNOWN ERROR CODE n" to the message buffer, where n is the ASCII equivalent of the error code you specified in AC0.

Figure 2-23 shows the structure of the error codes in ERMES. Use this structure for your own error message file if you want to use ?ERMSG to read it.

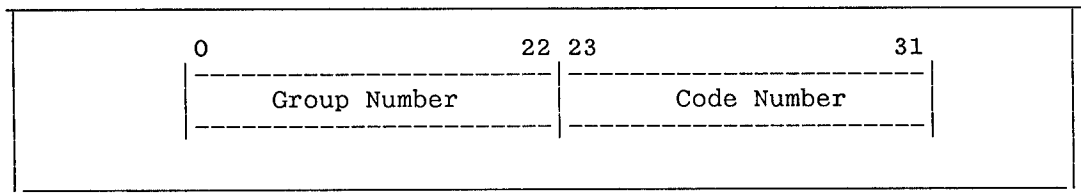


Figure 2-23. Error Code Structure in ERMES File

As Figure 2-23 shows, the error code entries in ERMES consist of two fields: group number, and code number. All error codes are categorized into groups ranging from 0 through octal 17777. Although groups 0 through octal 77 and octal 200 through octal 7777 are reserved for system definition, you can define error codes for groups octal 100 through octal 177 and octal 10000 through octal 17777.

You can change the error codes and text strings that your program reports. One important step is to edit file LINK_ERMES.CLI. See the documentation in file ERMES.SR, whose full pathname is usually :UTIL:ERMES.SR.

?ERMSG Continued

Two MASM macros, GRP and CODE, allow you to add new code groups and/or single codes to the error message file. The GRP macro defines the code group and the number of codes within that code group. The syntax of this macro is

GRP m n

where:

- m is the code group number.
- n is the maximum number of error codes within the code group.

For example,

```
GRP 100 200
```

defines code group 100, which can have up to 200 individual error codes.

The CODE macro inserts error codes and their associated text strings into an error code group. The syntax of the CODE macro is

CODE code-mnemonic .TXT *message-string*

For example,

```
CODE  ERFTM .TXT  *FILE/TAPE DENSITY MISMATCH*
```

associates the message string FILE/TAPE DENSITY MISMATCH with the error code ERFTM.

?ERSCH

Enables multitask scheduling for the calling process.

?ERSCH
error return
normal return

Input

None

Output

None

Error Codes in AC0

No error codes are currently defined.

Why Use It?

If a task disables scheduling with ?DRSCH, it has only two options to re-enable scheduling: the task can either ?KILL itself or it can issue ?ERSCH. Thus, you can use ?ERSCH to re-enable scheduling and maintain the calling task.

What It Does

?ERSCH re-enables multitask scheduling for the calling process. After the operating system executes ?ERSCH, it passes control to the normal return, and resumes its regular scheduling activities. The calling task may or may not resume execution, however, depending on its priority. If scheduling has already been enabled, it remains enabled, and ?ERSCH takes the normal return.

Notes

- See the descriptions of ?DRSCH, ?DFRSCH, and ?KILL in this chapter.

Licensed Material - Property of Data General Corporation

?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

Why Use It?

?ESFF makes sure that if the system fails, you do not lose any changes that you made to the shared file 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).

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.

?EXEC *[packet address]*
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	Some
Error Codes	None
Parameter Packet	Some

Input

AC0	Reserved (Set to 0.)
AC1	Host ID for Resource Management Agent (RMA) deflection (AOS/VS only), 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

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

Why Use It?

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

What It Does

?EXEC directs the EXEC utility to perform one of the following functions on behalf of the calling process:

- Assign or cancel a logical name to a tape unit or non-LD disk (an uninitialized disk that you want to treat as a whole unit) and issue an operator mount or dismount message.
- Mount an unlabeled tape. (See Figure 2-24 and Figure 2-25.)

Licensed Material - Property of Data General Corporation

- Mount a labeled tape. (See Figure 2–26 and Figure 2–27.)
- Dismount a labeled or unlabeled tape. (See Figure 2–28.)
- Place a request into a queue. (See Figure 2–29.)
- Hold, unhold, or cancel a queue request. (See Figure 2–30 and Figure 2–31.)
- Provide an EXEC status report. (See Figure 2–32 and Figure 2–33.)
- Submit a job to a MOUNT queue. (See Figure 2–34.)
- Dismount a unit (extended request). (See Figure 2–35.)
- Modify the queuing parameters of a queued job. (See Figure 2–36.)
- Obtain one or more queue names. (See Figure 2–37.)
- Obtain QDISPLAY information, given a queue name. (See Figure 2–38.)

Each of these functions takes a unique packet. However, the first offset, ?XRFNC, which defines the action to be performed, is common to all the packets. The second offset, ?XPRV, is also common; it contains the packet revision number.

*

Mounting Unlabeled and Labeled Tapes

If you use ?EXEC to mount a tape (labeled or unlabeled), you must specify an operator message string somewhere in your program.

Offsets ?XMUT and ?XMLT are byte pointers to the operator message text string. The operator message cannot exceed 80 characters, including the delimiter. The delimiter can be NEW LINE, carriage return, form feed, or null.

The operator message should contain at least some of the following information:

- The name of the tape reel (if it is not the volume name)
- Where the tape reel is stored
- The type of tape unit to use with the tape reel (for example, high–density, 7–track, or 9–track)
- Any special instructions to the operator

The logical name of the tape, to which offset ?XMUL or ?XMLL points, must be a simple unique name. EXEC takes this name and creates a link in the caller's initial working directory (as specified in the user's profile) with the pathname logon-dir:<LOGICALNAME>. This link resolves to @MTXn for unlabeled tape mounts and to @LMT:VOLID for labeled tape mounts. The X of @MTXN can be A, B, C, or D depending on the tape drive.

To mount an unlabeled tape (called a unit mount), you must set the first word of the ?EXEC packet, ?XRFNC, to ?XFMUN or ?XFXUN. (See Figure 2–24 and Figure 2–25.)

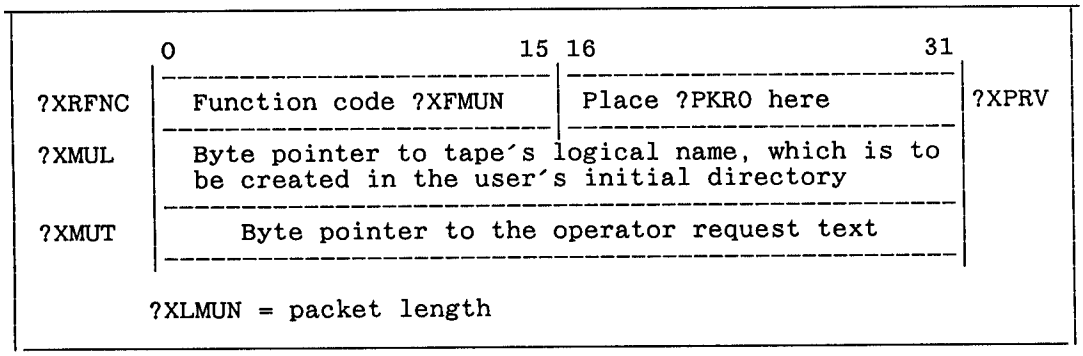


Figure 2-24. Structure of ?EXEC Packet for Unlabeled Mount Function ?XFMUN

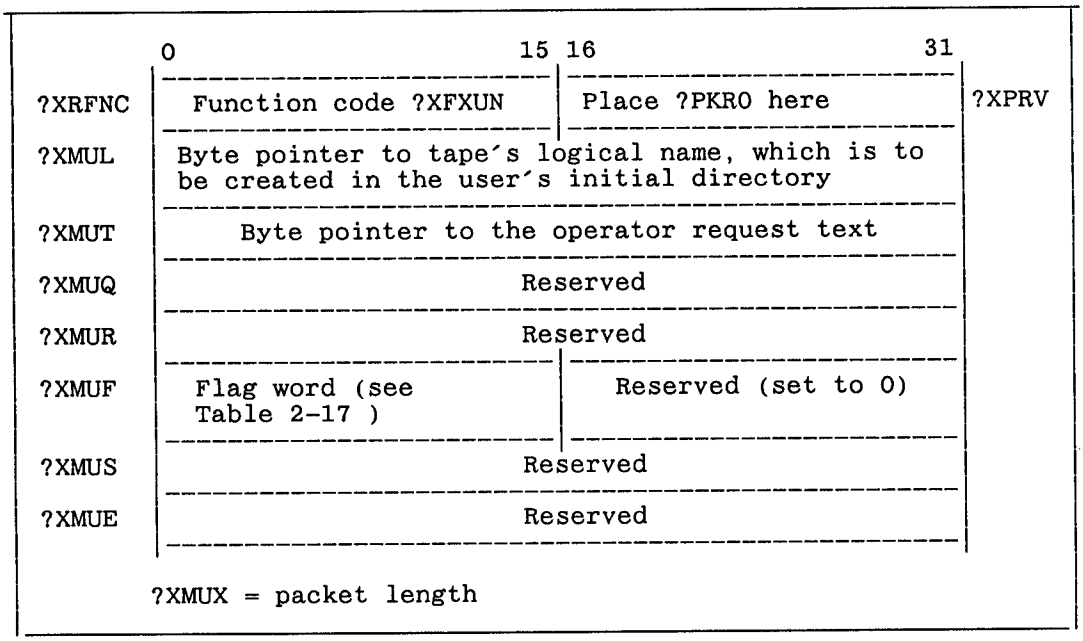


Figure 2-25. Structure of ?EXEC Extended Packet for Unlabeled Mount Function ?XFXUN

?EXEC Continued

To mount a labeled tape, you must set ?XRFNC to ?XFMLT or ?XFXML. (See Figure 2–26 and Figure 2–27.)

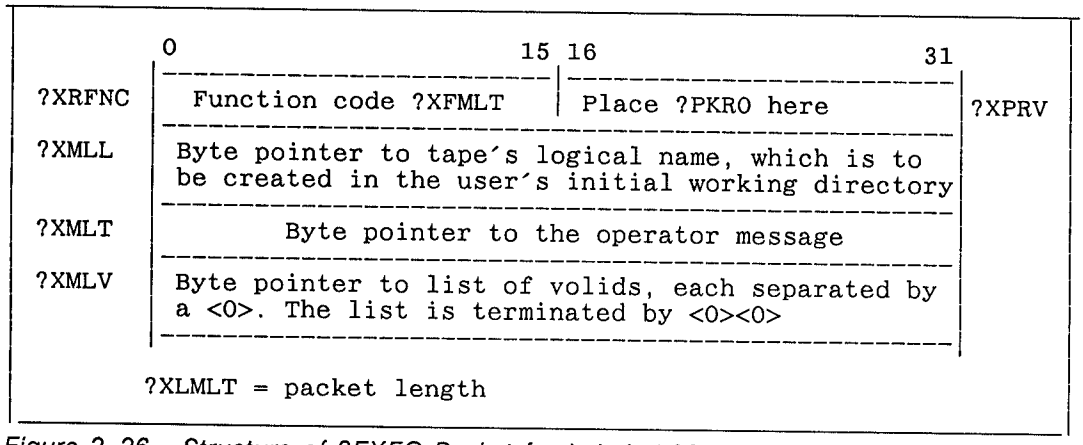


Figure 2–26. Structure of ?EXEC Packet for Labeled Mount Function ?XFMLT

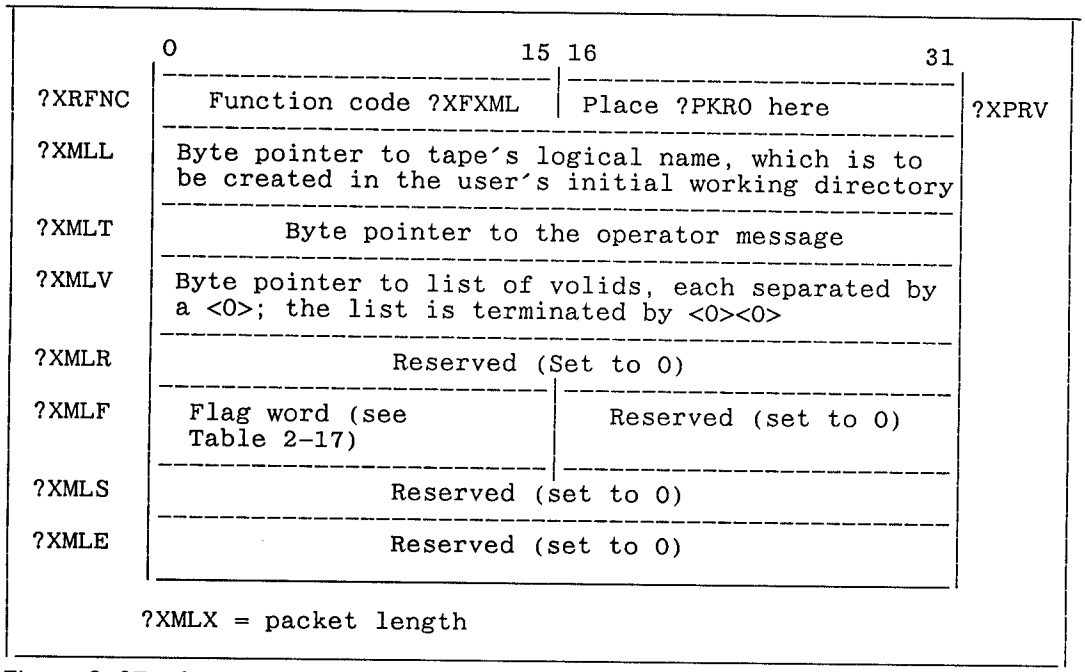


Figure 2–27. Structure of ?EXEC Extended Packet for Labeled Mount Function ?XFXML

Licensed Material – Property of Data General Corporation

Table 2-17. Flags for EXEC Functions ?XFXUN and ?XFXML

Flag	Description
?XMFC	If set, checks volid that operator specified against tape.
?XMFI	If set, the tape is in IBM format.
?OPD0 ?OPD1 ?OPD2	Density to use. This applies to tape drives capable of software selection of the density only. (See the description of the ?OPEN system call for information on setting the bits.)
?XMFR	If set, EXEC assumes that the tape to be mounted is to be read only and sets the ACL to USER,RE.
?XMEL	If set, allows you to extend the volid list. This applies to labeled tape mount requests only.

Dismounting Unlabeled and Labeled Tapes

To dismount either type of tape, you must set ?XRFNC to ?XFDUN. (See Figure 2-28.)

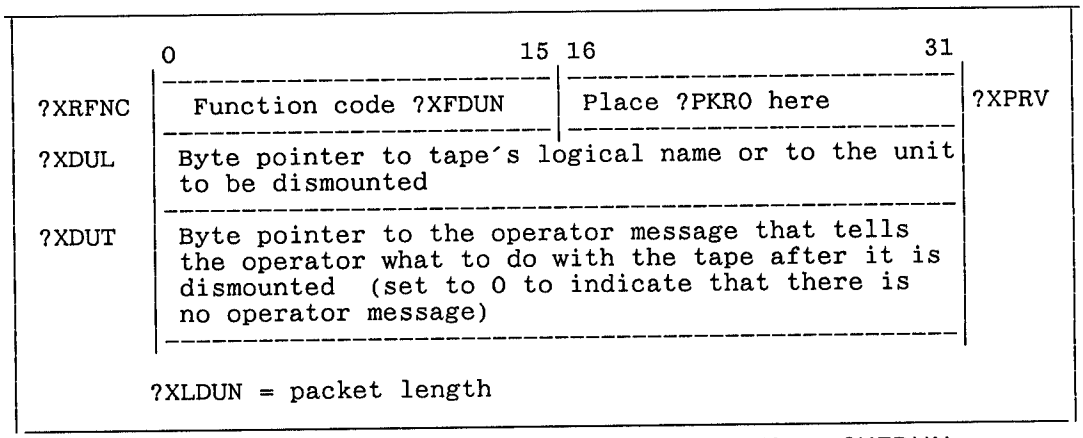


Figure 2-28. Structure of ?EXEC Packet for Dismounting a Tape, ?XFDUN

If you use ?EXEC to dismount a tape, an operator message is optional. (See "Mounting Unlabeled and Labeled Tapes" in this description for more information on operator messages.)

The number of characters in the operator message pointed to by offset ?XDUT cannot exceed 80 characters, including the terminator. (This is also true of the message pointed to by offset ?XMUT for unlabeled tapes. See Figure 2-24 and Figure 2-25.)

?EXEC Continued

Queuing a File Entry

The ?EXEC packet for placing a queue request allows you to queue a file for batch processing or for spooled output to one of the following devices:

- Line printer
- Console and other communications devices
- Digital plotter
- File transfer agent (FTA)
- SNA

Figure 2–29 shows the structure of the ?EXEC packet for queue requests, and Table 2–18 describes the contents of each offset. Note that some of the packet options correspond to CLI commands for the EXEC utility.

The first offset in the packet, ?XRFNC, defines the kind of queue. Other offsets allow you to postpone processing the queue request, assign the job an EXEC priority, delete the file after processing, and select several print options (for example, print the file's pathname, print page numbers, fold long lines, and so forth).

Offset ?XTYP points to the queue name of the desired queue. This name must correspond to the queue type chosen in offset ?XRFNC, unless you selected ?XFSUB (any type of queue) for the queue type.

Queue names resemble device names, but they do not begin with the @ prefix. For example, if you have one or more spooled line printers, one of the line printer queue names should be LPT. The system manager usually sets the queue names for an installation.

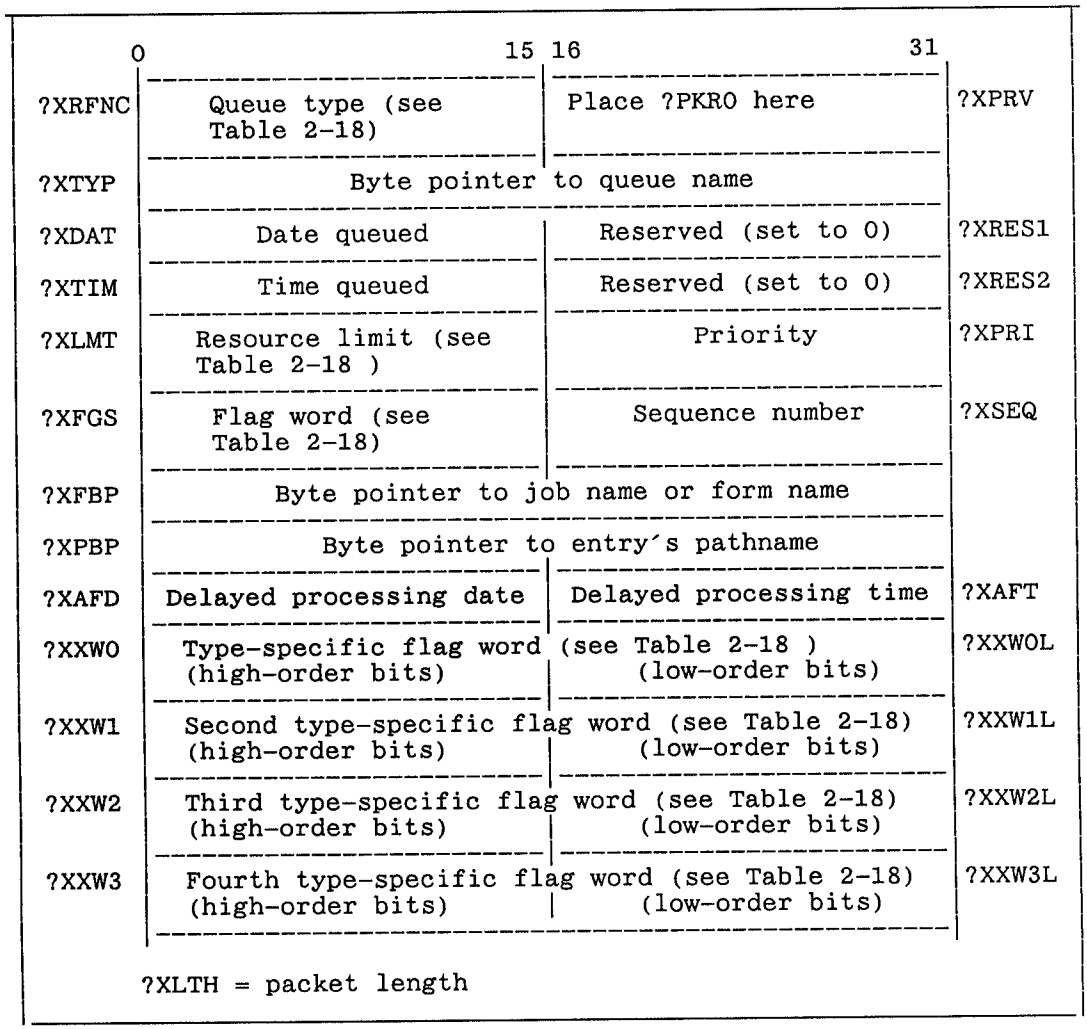


Figure 2-29. Structure of ?EXEC Packet for Queue Requests

Licensed Material - Property of Data General Corporation

Table 2-18. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XRFNC	Queue type: Place entry into one of the following: ?XFSUB--Any type of queue. ?XFLPT--Print queue. ?XFPLT--Plot queue. ?XFFTA--FTA queue. ?XFSNA--SNA/RJE queue. ?XFHAM--HAMLET queue. ?XFBAT--Batch queue. Setting Bit 0 of ?XRFNC indicates that RMA should deflect this request to the remote host specified in AC1 (AOS/VS only). 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.
?XPRV	Packet revision number. (Set to ?PKR0.)
?XTYP (double word)	Byte pointer to queue name (must correspond to type in ?XRFNC).
?XDAT	Date queued (returned by the OS in standard date notation).
?XRES1	Reserved for EXEC. (Set to 0.)
?XTIM	Time queued (returned by the OS in standard time notation).
?XRES2	Reserved for EXEC. (Set to 0.)
?XLMT	Resource limit (enforced only if operator enabled the EXEC limit feature): For print queue: ?XLMT = maximum number of pages to be printed. If you default, EXEC estimates the number of pages as follows: pages = (bytes/1000) + 4 where bytes = total number of bytes in file.

* There is no default unless otherwise specified.

(Continues)

Table 2-18. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XLMT	For batch queue: ?XLMT = maximum number of CPU (including the CLI and all other processes created by this batch job). If you default, EXEC estimates a value of 30 biseconds; that is, 1 minute. DEFAULT = 0 (see above)
?XPRI	Entry's priority, within the range: n (highest) through 255 (lowest). where n = maximum priority specified in your user profile. DEFAULT = -1 (EXEC assigns a priority in the middle of the range and returns the value here).
?XFGS	Flag word ?BMSH = 1B(?XFSH)--Set Sequence Number Held by User flag for this entry. ?BMDA = 1B(?XFDA)--Delete file specified in offset ?XPBP after processing. (Caller must have Owner or Write access to the file.) ?BMNR = 1B(?XFNR)--Do not restart entry if EXEC or the system crashes during processing. ?BMBI = 1B(?XFBI)--Output is in binary. ?BMOP = 1B(?XFOP)--Start processing only if operator is on duty. (We recommend setting this flag for batch jobs that require tape mounting or dismounting.) ?BMNO = 1B(?XFNO)--Notify the user's console when this entry is completed. (The message is lost if you log off, and then log on again; there is no message if you are not a descendant of EXEC.) ?BMPE = 1B(?XFPE)--Pend the calling process until the operator acts on the request. ?BMPE = 0B(?XFPE)--Don't pend the calling process, but still receive confirmation when the operator acts on the request.

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

Licensed Material - Property of Data General Corporation

Table 2-18. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XFGS (Continued)	?BMRA = 1B(?XFRA)--Determines the value of offsets ?XAFD and ?XAFT in this packet. ?BMTI = 1B(?XFTI)--Print titles on output listing. ?BMFO = 1B(?XFFO)--Fold long lines. DEFAULT = 0 (no flags).
?XSEQ	Sequence number EXEC assigned to this entry (returned by EXEC).
?XFBP (double word)	For batch queue: byte pointer to job name. For other queue type: byte pointer to form name. If you set this to 0, EXEC will use the device's default forms. (You must have previously defined the form filename and must have access to it.)
?XPBP (double word)	Byte pointer to full pathname (starting with the root) of the file to be processed. For FTA queues: byte pointer to full pathname of the source file.
?XAFD and ?XAFT	Interpretation depends on flag bit ?BMRA in offset ?XFGS. If ?BMRA was not set: ?XAFD - read as the date (in file system format) after which the entry can be processed. ?XAFT - read as the time (in file system format) after which the entry can be processed. If ?BMRA was set ?XAFD - read as a number of whole days after which the entry can be processed. ?XAFT - read as a number of bi-seconds (2-second intervals) after which the entry can be processed. If ?BMRA was set, EXEC adds the time periods specified in ?XAFD and ?XAFT, starting from the time the queue request was made, and begins processing the entry after the designated time period has elapsed. The default value for ?XAFD and ?XAFT is 0 (no delayed processing).

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

Table 2-18. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XXW0 (sometimes double word)	Type-specific flag word For print queue (single word): - If ?XXW0 is nonzero, start printing on the page number given here. - If ?XXW0 is 0 (the default), start printing on the first page. For batch queue (double word): - ?XXW0 is the high-order portion of a byte pointer to the pathname of the file you want to use as the @OUTPUT file. This file will be created if it does not already exist (EXEC does not print or delete this file). - If XXW0 is 0, EXEC creates, prints, and then deletes the Default Generic Output File, :QUEUE:USER.OUTPUT.SEQ. (If this file is empty, printing is omitted.) For FTA queue (single word): stream number if desired, otherwise 0. For all other queue types (single word): ?XXW0 must be 0. DEFAULT = 0 (no flags).
?XXW0L	For batch queue: ?XXW0L contains the low-order bits of ?XXW0. For all other queue types (single word): ?XXW0L must be 0. DEFAULT = 0.

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

Licensed Material - Property of Data General Corporation

Table 2-18. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XXW1 (sometimes double word)	Second type-specific flag word. For print queue (single word): If ?XXW1 is nonzero, stop printing at the page specified here. For batch queue (double word): ?XXW1 is a byte pointer to generic @LIST file; otherwise, contains 0 (for default @LIST file). For FTA queue (double word): byte pointer to full pathname of the destination file. For all other queue types (single word): ?XXW1 must be 0.
?XXW1L	For batch and FTA queues: ?XXW1L contains the low-order bits of ?XXW1. For all other queue types: ?XXW1L must be 0.
?XXW2	Third type-specific flag word. For batch queue: ?XXW2 must be 0. For SNA queue: Indicate the type of read by choosing one of the following mutually exclusive flags: ?X2LN - data-sensitive line read. ?X2SE - dynamic (sequential) read. ?X2EB - dynamic read of 80 characters. Optionally, indicate data translation: ?X2HO - Hollerith to EBCDIC translation. Specify the class of device with one of the following (these are mutually exclusive): ?X2CD - card. ?X2CN - console. ?X2EX - exchange.

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

Licensed Material - Property of Data General Corporation

Table 2-18. Contents of ?EXEC Packet for Queue Requests*

Offset	Contents
?XXW2 (Continued)	For FTA queue: Set one or more of the following transfer option flags: ?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. For all other queue types: ?XXW2 equals number of copies the EXEC spooler creates. (If ?XXW2 equals 0, one copy is assumed.)
?XXW2L	Reserved. (Set to 0.)
?XXW3 (sometimes double word)	For print and batch queues (double word): ?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. For SNA queue, set one of these mutually exclusive flags: ?X3TR - transparent transmission. ?X3CO - concatenate. For all other queue types (single word): ?XXW3 must be 0.
?XXW3L	For print and batch queues: ?XXW3L contains the low-order bits of ?XXW3. For all other queue types: ?XXW3L must be 0.

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

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.

?EXEC Continued

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.

Form name is the name of a form in the :UTIL:FORMS directory that was created with the CLI forms control utility (FCU). Typically, the system manager creates files in :UTIL:FORMS. Like the job name, the form name must contain from 1 through 31 legal filename characters. To determine the valid form names for your installation, type

```
ACL/V :UTIL:FORMS:+
```

This returns the names of the form files in :UTIL:FORMS and their corresponding ACLs. You can use any form file for which you have Read access.

Offsets ?XAFD and ?XAFT allow you to delay processing the queue request until a specific time has elapsed. The interpretation of these offsets varies, depending on whether you set flag ?XFRA in offset ?XFGS.

If you set the ?XFRA flag, you must set offsets ?XAFD and ?XAFT to the desired date and time offsets from the current time. If you do not set the ?XFRA flag, set offsets ?XAFD and ?XAFT to the desired processing date and time in the standard notations.

Offset ?XXW2 is a print flag with different meanings for different queue types. For a batch queue, this flag must be 0. For an FTA queue you can select any combination of transfer options flags. An SNA queue has a number of options associated with it, several of which are mutually exclusive. You can choose ONE read type (data-sensitive, dynamic, or dynamic 80-character) and ONE class of device (card, console, or exchange). You also have the option of indicating Hollerith to EBCDIC translation with the ?X2HO flag.

For any other queue, ?XXW2 is the number of copies the EXEC spooler creates (0 is interpreted as one copy.)

Using Hold, Unhold, Cancel Queue Requests (AOS/VS)

You can also use ?EXEC to suspend (hold), release (unhold), or clear (cancel) a queue entry you have created. An unhold directive negates a previously issued hold directive. Figure 2–30 shows the structure of the AOS/VS packet for the hold, unhold, and cancel directives. Table 2–19 lists the AOS/VS ?EXEC packet's contents.

As in the other ?EXEC packets, the value of offset ?XRFNC determines the action you want to perform: ?XFHOL holds the specified entry, ?XFUNH releases the entry, and ?XFCAN cancels the entry. In addition to the ?XFHOL specification, each queue request has one hold bit that only the operator process can access. If any hold bit is set, EXEC does not select the entry for processing.

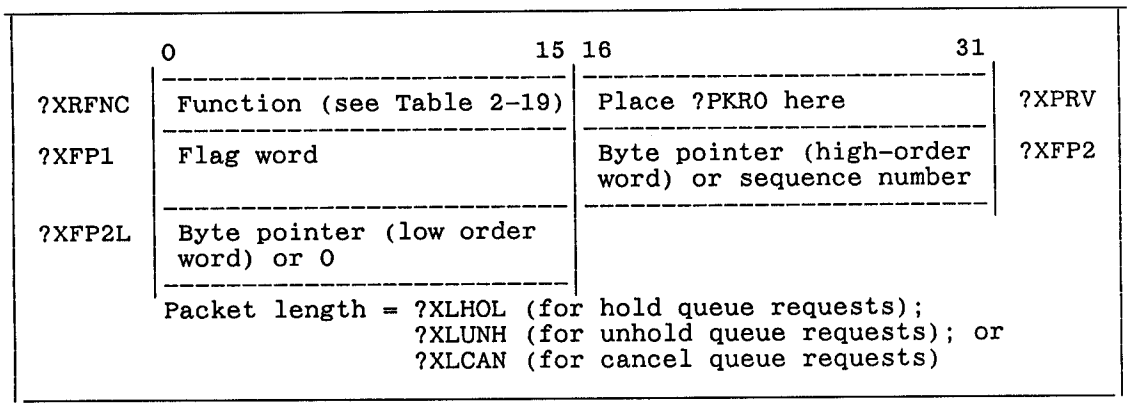


Figure 2–30. Structure of AOS/VS ?EXEC Packet for Hold, Unhold, or Cancel Queue Requests

Offset ?XFP2 contains either the high-order word of a byte pointer to the entry's job name or else the entry's sequence number.

If you use offsets ?XFP2/?XFP2L as a byte pointer, EXEC searches only the batch queue, and changes only the status of the jobs which have this job name. If these offsets contain a sequence number (?XFP2L contains zero), EXEC searches all queue entries and changes the hold bit of the entry designated by the specified sequence number.

When you select the Cancel option (?XFCAN), EXEC clears all hold bits, including the operator bits, and sets the entry's Canceled by User bit.

Table 2-19. Contents of AOS/VS ?EXEC Packet for Hold, Unhold, or Cancel Queue Requests*

Offset	Contents
?XRFNC	Function: select one of the following values: ?XFHOL--Suspend (hold) the specified entry. ?XFUNH--Release (unhold) the specified entry. ?XFCAN--Clear (cancel) the specified entry.
?XPRV	Packet revision number. Place ?PKR0 here.
?XFP1	Flag word - If ?XFP2 and ?XFP2L contain a byte pointer to job name, set to -1. - If ?XFP2 and ?XFP2L contain a sequence number, set to any value.
?XFP2 (double word)	Contains one of the following: - Byte pointer to job name. - Sequence number. (Set ?XFP2L to 0.)

* There is no default unless otherwise specified.

Job names and sequence numbers need not be unique; that is, it is possible to have several requests with the same job name or sequence number. In this case, EXEC performs the hold, unhold, or cancel action on all entries that bear the specified sequence number or job name.

Note that the hold and unhold directives do not affect an entry that is currently being processed. The cancel directive will stop the processing of a currently active request.

Using Hold, Unhold, Cancel Queue Requests (AOS/DVS)

You can also use ?EXEC to suspend (hold), release (unhold), or clear (cancel) a queue entry you have created. An unhold directive negates a previously issued hold directive. Figure 2-31 shows the structure of the AOS/DVS packet for the hold, unhold, and cancel directives. Table 2-120 lists the AOS/VS ?EXEC packet's contents.

As in the other ?EXEC packets, the value of offset ?XRFNC determines the action you want to perform: ?XFHOL holds the specified entry, ?XFUNH releases the entry, and ?XFCAN cancels the entry. In addition to the ?XFHOL specification, each queue request has one hold bit that only the operator process can access. If any hold bit is set, EXEC does not select the entry for processing.

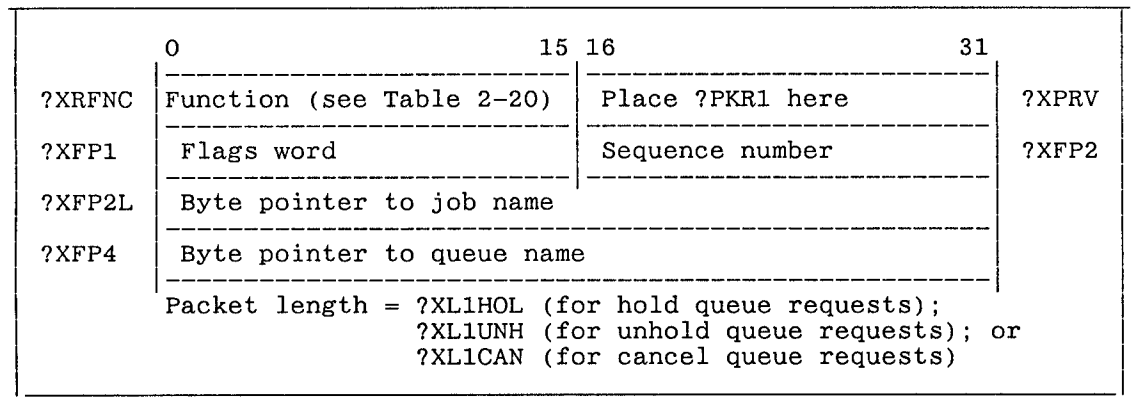


Figure 2-31. Structure of AOS/DVS ?EXEC Packet for Hold, Unhold, or Cancel Queue Requests

If you place 0 in offset ?XFP2 and a byte pointer in offset ?XFP2L, EXEC searches only the batch queue, and changes only the status of the jobs which have this job name. If you place a nonzero number in offset ?XFP2 and zero in offset ?XFP2L, EXEC searches all queue entries and changes the hold bit of the entry designated by the specified sequence number.

When you select the Cancel option (?XFCAN), EXEC clears all hold bits, including the operator bits, and sets the entry's Canceled by User bit.

Table 2-20. Contents of AOS/DVS ?EXEC Packet for Hold, Unhold, or Cancel Queue Requests*

Offset	Contents
?XRFNC	Function: select one of the following values: ?XFHOL--Suspend (hold) the specified entry. ?XFUNH--Release (unhold) the specified entry. ?XFCAN--Clear (cancel) the specified entry.
?XPRV	Packet revision number. Place ?PKR1 here.
?XFP1	Flag word - If ?XFP2L contains a byte pointer to job name, set to -1. - If ?XFP2 contains a nonzero number, set to any value.
?XFP2	Sequence number of the queue entry you want to hold, release, or cancel. If you want to identify the queue entry by its job name, supply 0 here and a byte pointer to the job name in offset ?XFP2L.
?XFP2L (double word)	If ?XFP2 contains a positive number, set to 0. If ?XFP2 contains 0, supply a byte pointer to the job name. If ?XFP2L contains a byte pointer to a null string, all jobs you have submitted will be affected (i.e., held, released, or cancelled).
?XFP4 (double word)	Along with sequence number/job name, you can specify a queue name. Doing so allows deflection of your request to the EXEC process that handles the queue you identify in this offset.

* There is no default unless otherwise specified.

Job names and sequence numbers need not be unique; that is, it is possible to have several requests with the same job name or sequence number. In this case, EXEC performs the hold, unhold, or cancel action on all entries that bear the specified sequence number or job name.

Note that the hold and unhold directives do not affect an entry that is currently being processed. The cancel directive will stop the processing of a currently active request.

Obtaining EXEC Status Information

To get selected information about the EXEC process that is currently running, issue ?EXEC with the packet shown in Figure 2-32. The operating system returns the status information to offset ?XFP1 of the packet and, optionally, to the buffer you specify in offset ?XFP2. Table 2-21 describes each offset in the packet.

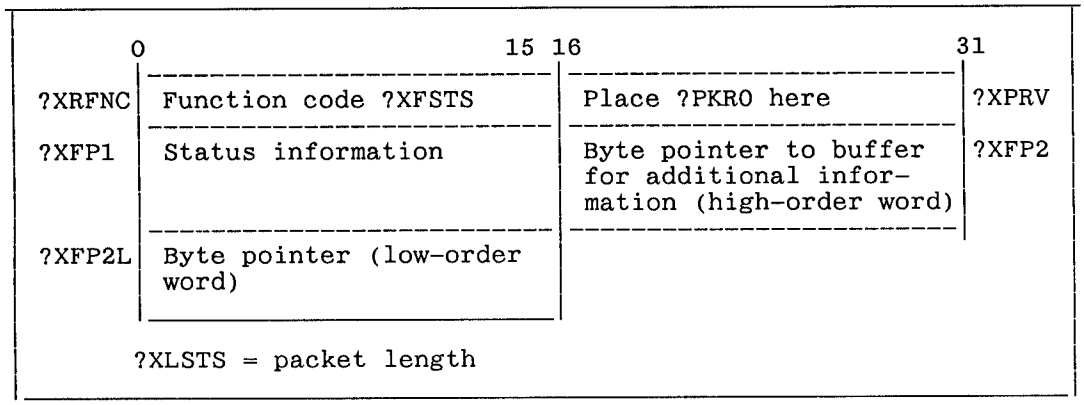


Figure 2-32. Structure of ?EXEC Packet for Status Information

Table 2-21. Contents of ?EXEC Packet for Status Information*

Offset	Contents
?XRFNC	?XFSTS (specifies function: obtain EXEC status information).
?XPRV	Packet revision number. (Set to ?PKR0.)
?XFP1	Status information (returned by EXEC): - If Bit 0 = 1, operator is on duty. - If Bit 0 = 0, system is unattended. - If Bit 1 = 1, your username is logged on in batch mode. - If Bit 1 = 0, your username is logged on at a console or is not logged on. - If Bits 8-15 = 0, you are not logged on. - If Bits 8-15 = PID of the process immediately subordinate to EXEC, you are logged on.
?XFP2 (double word)	Byte pointer to buffer area you set aside to receive the console name or the batch stream name (if you were logged on under EXEC). (The OS returns the console name without the @ prefix.) DEFAULT = 0 (no buffer).

* There is no default unless otherwise specified.

Obtaining Extended EXEC Status Information

In addition to the information that function code ?XFSTS returns, you can also obtain the PID of the EXEC process. You do this by specifying function code ?XFXTS in offset ?XRFNC. Issue ?EXEC with the packet shown in Figure 2-33. Table 2-22 describes each offset in the packet.

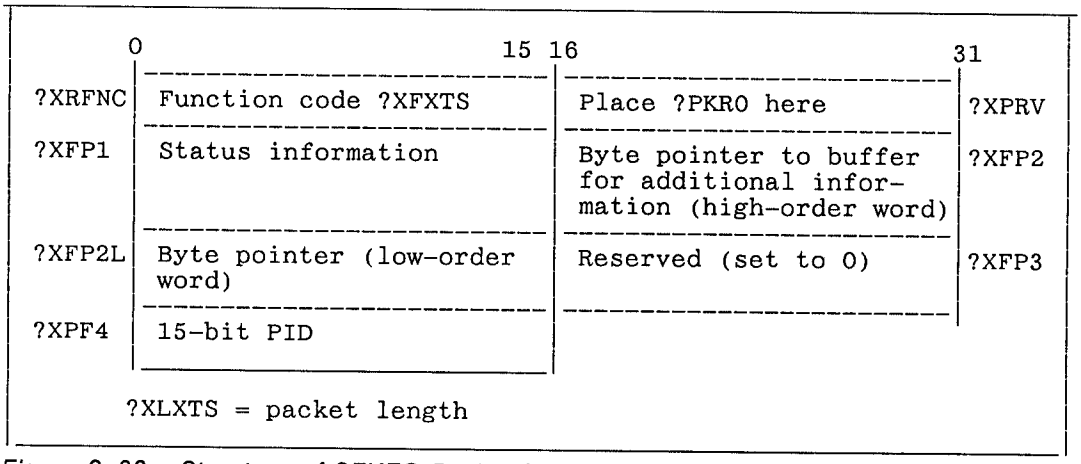


Figure 2-33. Structure of ?EXEC Packet for Extended Status Information

Table 2-22. Contents of ?EXEC Packet for Extended Status Information*

Offset	Contents
?XRFNC	?XFXTS (specifies function: obtain EXEC extended status information).
?XPRV	Packet revision number. (Set to ?PKR0.)
?XFP1	Status information (returned by EXEC): - If Bit 0 = 1, operator is on duty. - If Bit 0 = 0, system is unattended. - If Bit 1 = 1, your username is logged on in batch mode. - If Bit 1 = 0, your username is logged on at a console or is not logged on. - If Bits 8-15 = 0, you are not logged on. - If Bits 8-15 = PID of the process immediately subordinate to EXEC, you are logged on.
?XFP2 (double word)	Byte pointer to buffer area you set aside to receive the console name or the batch stream name (if you were logged on under EXEC). (The OS returns the console name without the @ prefix.) DEFAULT = 0 (no buffer).
?XFP3	Reserved. (Set to 0.)
?XFP4	The OS returns the 15-bit PID of the EXEC process that is currently running.

* There is no default unless otherwise specified.

Submitting a Job to a MOUNT Queue

You use the ?XFMNT function of ?EXEC to submit a job to a MOUNT queue. The CLI MOUNT command uses the function, and it allows specification of queueing parameters such as /HOLD and /AFTER. If you set the ?XFPE flag bit, it indicates that the process should be pended until the operator responds to the queue request (e.g., mounts the tape or refuses the request). If you don't set the ?XFPE flag bit, the process takes the normal ?EXEC call return as soon as the request becomes part of the queue. An operator's response to an unpended mount request (i.e., the ?XFPE flag bit is off) is relayed to the caller so that the caller is aware of the operator's actions.

The ?XFMNT packet in Figure 2-34 allows specification of all the ?XFMUN and ?XFMLT options plus some of the ?XFSUB options. The details of these three options appear respectively in Figure 2-24, Figure 2-26, and Table 2-18/Figure 2-29. You can use the ?XWW1 offset in Figure 2-34 to specify a tape or diskette mount. If you specify neither type of mount, ?EXEC assumes you are issuing a tape mount request.

You can specify a directory location for the link name. In this case, the operating system places in offset ?XWW3 a byte pointer. This byte pointer points to the directory name where the logical name will be created.

Issue ?EXEC with the packet shown in Figure 2-34.

	0 15	16 31	
?XRFNC	Function code ?XFMNT	Place ?PKR0 here	?XPRV
?XTYP	Byte pointer to queue name (-1 if the default)		
?XDAT	Date queued (returned)	Additional flags (set to 0)	?XFG2
?XTIM	Time queued (returned)	Reserved (set to 0)	?XRES2
?XLMT	Reserved (set to 0)	Reserved (set to 0)	?XPRI
?XFGS	Flags word	Sequence number (returned)	?XSEQ
?XFBP	Byte pointer to VOLID list		
?XPBP	Byte pointer to logical name		
?XAFD	/AFTER date, else 0	/AFTER time, else 0	?XAFT
?XWW0	Mount status bits	Reserved (set to 0)	?XWW0L
?XWW1	Media type (tape or diskette)	Reserved (set to 0)	?XWW1L
?XWW2	Byte pointer to operator message		?XWW2L
?XWW3	Byte pointer to directory pathname for link, else 0		?XWW3L
	?XLMNT = packet length		

Figure 2-34. Structure of ?EXEC Packet for a MOUNT Queue Request

Dismounting a Unit (Extended Request)

You can specify a directory where the logical name of the dismounted unit will be located. You must also specify this directory pathname at DISMOUNT time.

Under AOS/DVS only, you can specify a registered queue name as the target of a mount submission. If this queue name is served on a remote host, you must similarly deflect the dismount that the remote host will service. To accomplish this, the packet is extended so it contains a queue name specification area in offset ?XXDQ.

Issue ?EXEC with the packet shown in Figure 2-35.

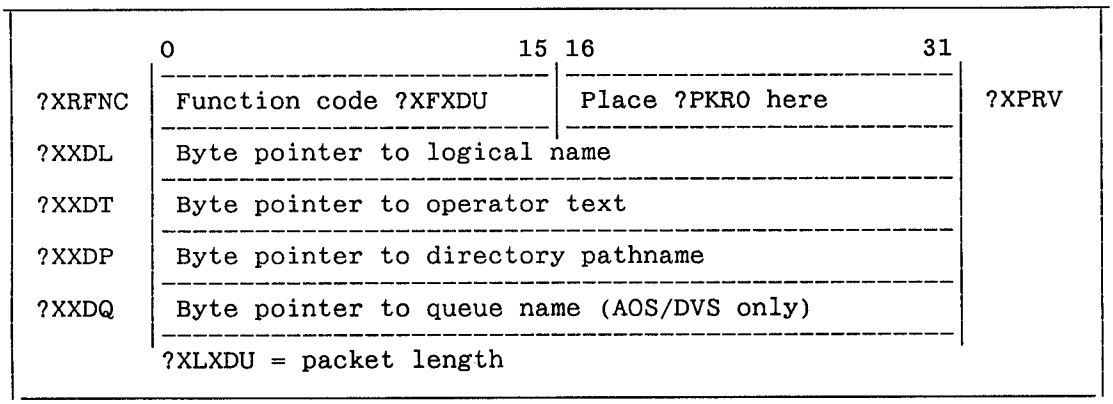


Figure 2-35. Structure of ?EXEC Packet for Dismounting a Unit (Extended Request)

Modifying the Queuing Parameters of a Queued Job

You use the ?XFMOD function of ?EXEC to request a change in the queuing parameters of a job that you previously queued. The packet in Figure 2-36 shows the structure of a packet for the ?XFMOD function. Offsets whose areas begin with an asterisk (such as ?XFBP) are offsets whose values you can change. You can also reset flags with this version of the ?EXEC call.

Note the similarity of the packet in Figure 2-36 with the submission-type packets such as the one in Figure 2-34.

Offset ?XTYP differs between AOS/VS and AOS/DVS.

Table 2-23 gives more information about offsets ?XWM0 through ?XWM3 in Figure 2-36.

	0 15	16 31	
?XRFNC	Function code ?XFMOD	Place ?PKR0 here	?XPRV
?XTYP	-1 under AOS/VS; byte pointer to registered queue name or -1 under AOS/DVS		
?XDAT	Not used (set to 0)	* Additional flags	?XFG2
?XTIM	Not used (set to 0)	Not used (set to 0)	?XRES2
?XLMT	* New /CPU value or new maximum pages value	* New /QPRIORITY value	?XPRI
?XFGS	* Flags word	Target sequence number for entry to be modified	?XSEQ
?XFBP	* Byte pointer to new job name or form name		
?XPBP	Not used (set to 0)		
?XAFD	* New /AFTER date	* New /AFTER time	?XAFT
?XWW0	* New XW0 value		?XWW0L
?XWW1	* New XW1 value		?XWW1L
?XWW2	* New XW2 value		?XWW2L
?XWW3	* New XW3 value		?XWW3L
?XUSR	Not used (set to 0)		
?XPWD	Not used (set to 0)		
?XHBP	* Byte pointer to new header value		
?XMBP	* Byte pointer to new mapper filename		
?XWM0	Modify flags	Modify flags	?XWM1
?XWM2	Not used (set to 0)	Not used (set to 0)	?XWM3
	?XLMOD = packet length		

Figure 2-36. Structure of ?EXEC Packet for Changing Queuing Parameters

Required fields in the QMODIFY packet of Figure 2–36 are:

- Offset ?XRFNC must contain function code ?XFMOD.
- Offset ?XPRV must contain value ?PKR0.
- Offset ?XTYP must contain –1 under AOS/VS. It can contain –1 under AOS/DVS. If the job being addressed is "remote," offset ?XTYP must specify a registered queue name.
- Offset ?XSEQ must contain the sequence number of the job you want to modify.

Set all unused offsets in the packet to zero. If you want to change any of the original queued parameters, place the new value in the appropriate location in the packet. This location is the same one you would specify if you were queuing the job for the first time.

The bit flags in offsets ?XWM0, ?XWM1, ?XWM2, and ?XWM3 appear in Table 2-23.

Table 2-23. Contents of Selected Offsets in the ?EXEC Packet for the ?XFMOD Function

Offset	Contents
?XWM0	Bit flags --------- 1B0 -- Reserved (Set to 0.) 1B1 -- Reserved (Set to 0.) 1B2 -- Reserved (Set to 0.) 1B3 -- Reserved (Set to 0.) 1B4 -- (/CPU) Offset ?XLMT specifies a new CPU limit. 1B5 -- (/BEGIN) Offset ?XWW0 specifies a new beginning page. 1B6 -- (/END) Offset ?XWW1 specifies a new ending page. 1B7 -- (/COPIES) Offset ?XWW2 specifies a new number of copies. 1B8 -- (/PAGES) Offset ?XLMT specifies a new page limit. 1B9 -- (/QPRIORITY) Offset ?XPRI specifies a new queue priority value. 1B10 -- (/JOBNAME=) Offset ?XFBP contains a byte pointer to a new jobname. 1B11 -- (/FORMS=) Offset ?XFBP contains a byte pointer to a new formname. 1B12 -- (/AFTER=) Offsets ?XAFD and ?XAFT contain a new /AFTER date and time. 1B13 -- (/QOUTPUT=) Offset ?XWW0 contains a byte pointer to a new queue output file. 1B14 -- (/QLIST=) Offset ?XWW1 contains a byte pointer to a new queue list file. 1B15 -- (/DEST=) Offset ?XWW3 contains a byte pointer to a new string (batch, print, plot only).
?XWM1	Bit flags --------- 1B0 -- (/HOLD) Examine bit ?XFSH in offset ?XFGS. 1B1 -- (/DELETE) Examine bit ?XFDA in offset ?XFGS. 1B2 -- (/RESTART) Examine bit ?XFNR in offset ?XFGS. 1B3 -- (/BINARY) Examine bit ?XFBI in offset ?XFGS. 1B4 -- (/OPERATOR) Examine bit ?XFOP in offset ?XFGS. 1B5 -- (/NOTIFY) Examine bit ?XFNO in offset ?XFGS. 1B6 -- Reserved (Set to 0.) 1B7 -- (/TITLES) Examine bit ?XFTI in offset ?XFGS. 1B8 -- (/FOLDLONGLINES) Examine bit ?XFFO in offset ?XFGS. 1B9 through 1B15 -- Reserved (Set to 0.)
?XWM2	Reserved (Set to 0.)
?XWM3	Reserved (Set to 0.)

Obtaining a List of Queue Names

You use the ?XFNQN function of ?EXEC to request a list of queue names. You can issue the call repeatedly to obtain names of all queues of a given type or of all queues regardless of type. Since this call can be iterative, we provide a "get next" key field in the parameter packet. This offset (the 64-bit field formed by ?XFHK and ?XFLK) has an initial value of 0. Successive iterations use and modify this offset. System calls ?GNFN, and ?EXEC with function code ?XFQDS, also use a "get next" key field.

Issue ?EXEC with the packet shown in Figure 2-37.

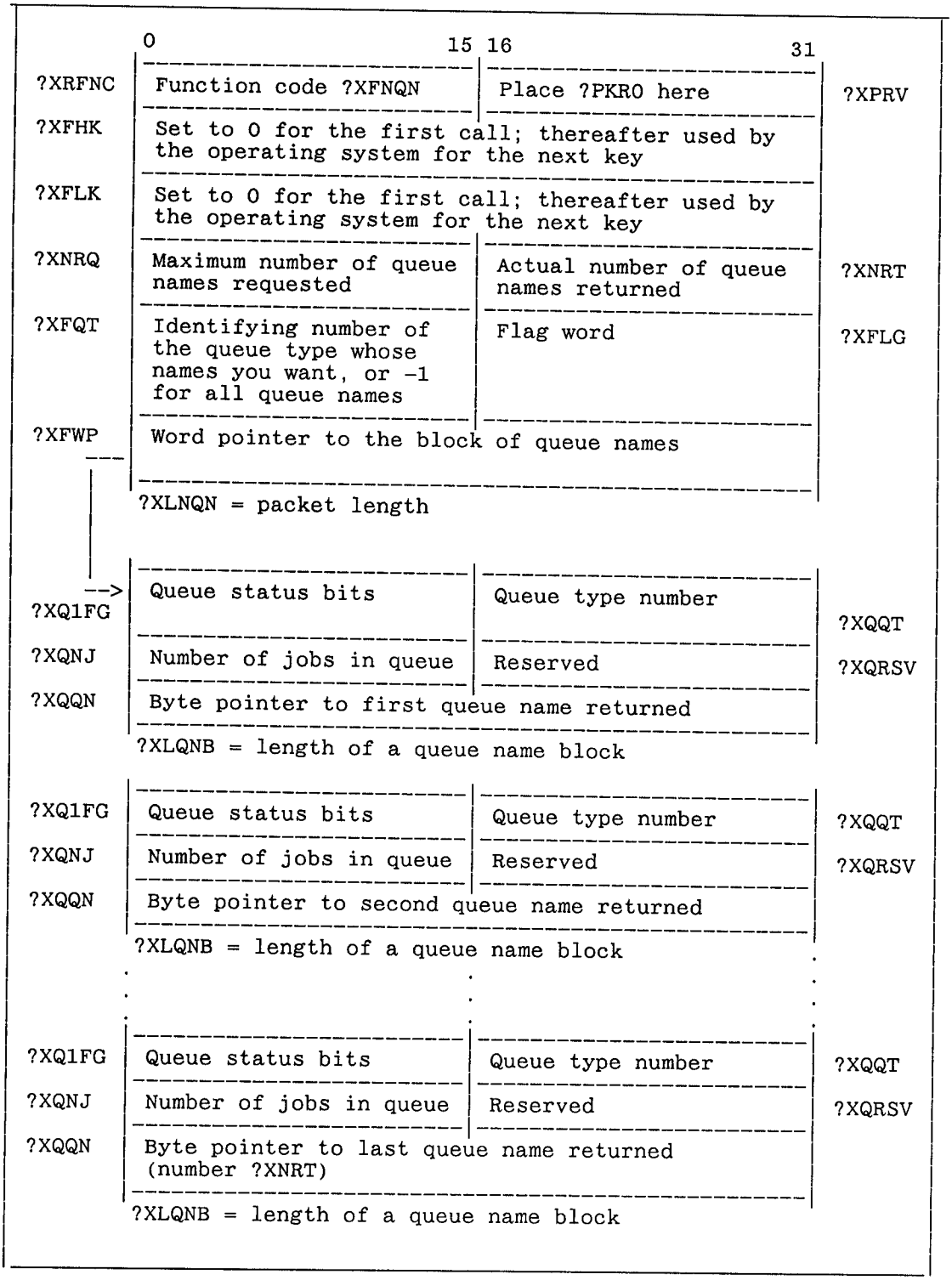


Figure 2-37. Structure of ?EXEC Packet for Obtaining Queue Names

The flag word in offset ?XFLG contains a bit that indicates whether or not a queue name was provided on input. If so, the byte pointer in the first queue name block points to the queue name. Offset ?XQ1FG contains a bit that indicates the open/closed status of the queue. Offset ?XQQT contains a numeric indicator of the queue type (PRINT, BATCH, FTA, MOUNT, etc.). PARU.32.SR defines these numeric values.

Obtaining QDISPLAY Information

You use the ?XFQDS function of ?EXEC to obtain QDISPLAY information. Each ?EXEC call of type ?XFQDS can obtain from 1 to 32 units of QDISPLAY information. You specify, in offset ?XFQN, the name of the queue for which you want QDISPLAY information.

Since this call can be iterative, we provide a "get next" key field in the parameter packet. This offset (the 64-bit field formed by ?XFHK and ?XFLK) has an initial value of 0. Successive iterations use and update this offset. System calls ?GNFN and ?EXEC (with function code ?XFNQN) also use a "get next" key field.

To obtain a complete list of all entries in a given queue, issue ?EXEC with key field ?XFHK/?XFLK set to zero. Then repeatedly reissue ?EXEC, without changing fields ?XFHK/?XFLK and ?XFQN, until error END OF QUEUE is returned. Repeat this process for each queue for which you need to obtain QDISPLAY information.

For the flag words in the buffers that receive the information

- The status flag word, offset ?XD3FG, has one bit defined. 0B0 represents an inactive job and 1B0 represents an active job.
- Bit definitions for the flag word, offset ?XD2FG, are the same as for those of ?XFG2 in queue submission packets.
- Bit definitions for the additional flag word, offset ?XDSFG, are the same as for those of ?XFGS in queue submission packets.

Issue ?EXEC with the packet shown in Figure 2-38.

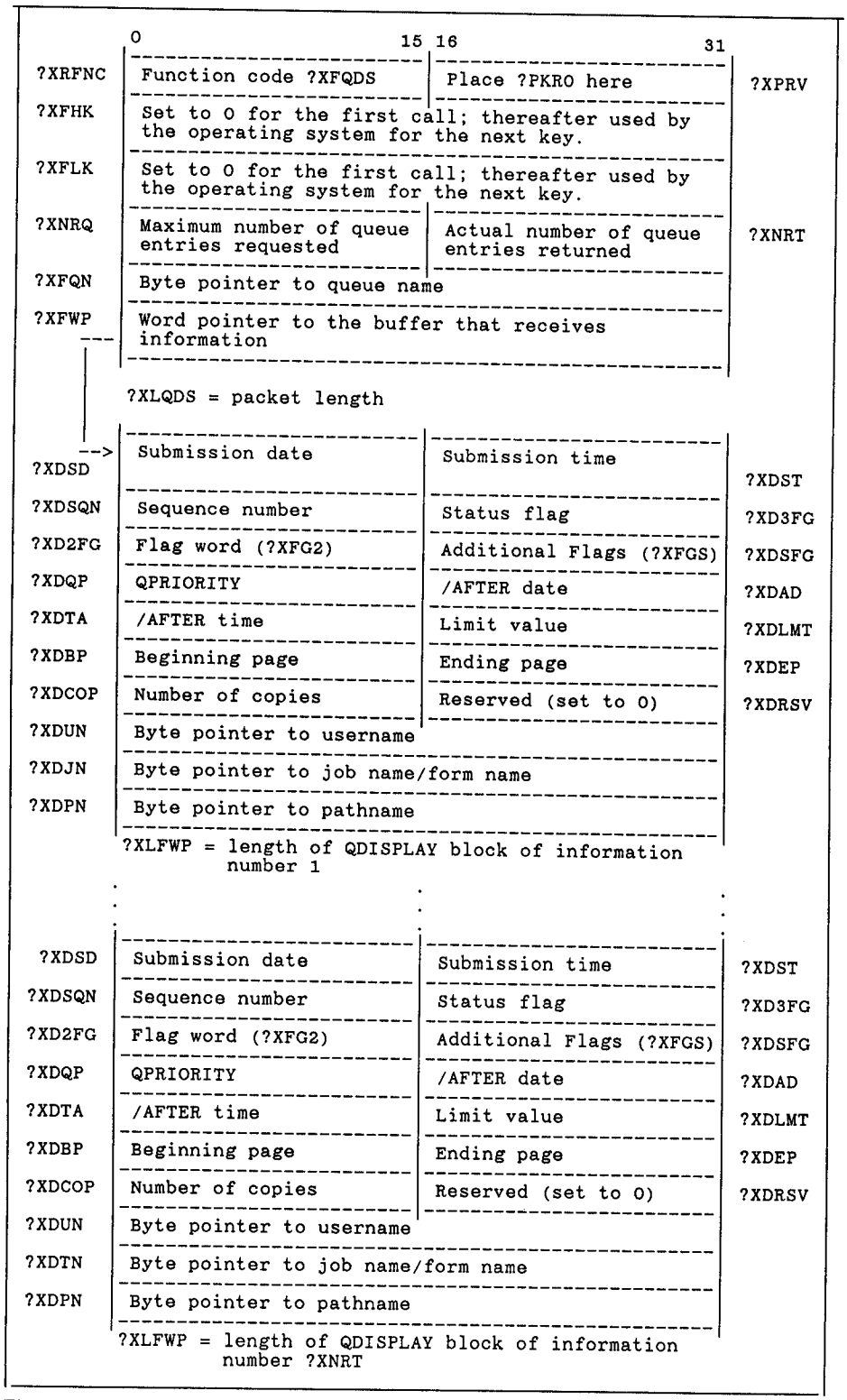


Figure 2-38. Structure of ?EXEC Packet for Obtaining QDISPLAY Information

Licensed Material - Property of Data General Corporation

Sample Packet

The following sample packet shows mounting an unlabeled tape:

```
PKT:    .BLK      ?XLMUN                ;Packet length is ?XLMUN.

        .LOC      PKT+?XRFNC            ;What are you going to do?
        .WORD     ?XFMUN                ;Mount an unlabeled tape.

        .LOC      PKT+?XRES             ;Reserved
        .WORD     0                     ;(Set to 0.)

        .LOC      PKT+?XMUL             ;Byte pointer to tape's
                                        ;logical name?

        .DWORD    LNAM*2                ;Byte pointer to LNAM.

        .LOC      PKT+?XMUT             ;Byte pointer to operator
                                        ;message?

        .DWORD    MSG*2                 ;Byte pointer to MSG.

        .LOC      PKT+?XLMUN            ;End of packet.

LNAM:   .TXT      /TAPE/                ;Logical tape name.

MSG:    .TXT      /TAPE IS ON FLOOR/    ;Operator text message.
```

Notes

- Refer to your operating system's "How to Generate and Run" manual for information on the EXEC commands, setting queue names, batch processing, and the format directory.

Licensed Material - Property of Data General Corporation

?EXPO

Sets, clears, or examines execute–protection status.

?EXPO
error return
normal return

Input

AC0	Starting address of the target region
AC1	Ending address of the target region
AC2	One of the following: • –1 to clear execute protection • 0 to set execute protection • Bit pointer to the start of a bit array that defines the region's execute–protection status

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERICM	Illegal system command (invalid system call for 16–bit processes)
ERPRE	Invalid system call parameter
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can prevent your program from executing certain logical pages, such as pages that contain data, by setting execute protection. This makes it easier to find errors in your code.

What It Does

Depending on your input to AC2, ?EXPO performs one of the following functions:

- Sets execute protection for the logical address region that you specify in AC0 and AC1.
- Clears execute protection from the logical address region that you specify in AC1.
- Returns the current execute-protection status for the region that you specify.

Setting and Clearing Execute Protection

To set or clear execute protection for a region, load AC0 with the starting address of the region and load AC1 with the ending address of the region before you issue ?EXPO. The starting and ending addresses must be in the same ring.

When you use ?EXPO to set execute protection, the operating system checks to see if the starting address is the first word on a page and if the ending address is the last word on a page. If either address falls on a page boundary, the operating system includes that page in the execute-protected region. If neither the starting nor the ending address falls on a page boundary, the operating system excludes that page from the execute-protected region.

Examining Execute-Protection Status

To find out whether the specified region is execute protected, perform the following steps:

1. Define a bit array in your address space.

   Bit 0 in this array represents the first page in the target region,
   Bit 1 represents the second page, and so on to the end of the array.

2. Load the array's bit pointer relative to the calling segment into AC2.

3. Issue ?EXPO.

On output, the operating system sets the corresponding bit to 1 if the page is execute protected; it sets the corresponding bit to 0 if the page is not execute protected. In other words, each bit that is set means that the page that that bit represents is execute protected.

?FDAY

Converts date to a scalar value.

?FDAY
error return
normal return

Input

AC0	Days from 1 through 31
AC1	Month from 1 through 12
AC2	Year minus 1900 from 0 through 2099

Output

AC0	Scalar value for days (the number of days that have elapsed since 31 December 1967)
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPRE	Invalid system call parameter

Why Use It?

?FDAY is a useful way to obtain input parameters for the ?CREATE system call's time block, an additional parameter block you can use to record a file's creation date and time, and the day and time it was last accessed or modified. The parameters for the ?CREATE time block must be scalar values.

What It Does

?FDAY is the inverse of ?CDAY; that is, it converts the date, expressed in standard notation, to a scalar value from a base date of 31 December 1967. Before you issue ?FDAY, load AC0, AC1, and AC2 with octal values for the day, month, and year, respectively. To obtain the input value for AC2, subtract 1900 from the year.

You can use any legal dates from 1 January 1968 (1,1,68) though 31 December 2099 (31,12,199) as input parameters for ?FDAY. To determine the day of the week from the output, divide the value in AC0 by 7 and note the remainder. Every date with a remainder of 1 (for example, 1 January 1968) fell on Monday.

Notes

- See the description of ?CREATE in this chapter for information on the ?CREATE time block.

?FEDFUNC Interfaces to File Editor (FED) utility.

?FEDFUNC
error return
normal return

Input

AC0	Unused
AC1	Unused
AC2	Address of packet

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERUFR	Unknown function request
ERVMP	Invalid address

Why Use It?

?FEDFUNC provides you with a simple-to-use interface to the File Editor (FED) utility.

?FEDFUNC Continued

What It Does

?FEDFUNC provides the caller with an interface to some of the FED utility routines. Each function takes a unique packet. However, the first offset, which defines the function, is common to all packets. The second offset is also common to all packets.

Here's the correspondence between figure numbers that show the structures of the packets and the functionality of the packets.

Figure 2–39	?FRCR	Change radix
Figure 2–40	?FROST	Open symbol table file
Figure 2–41	?FREFS	Evaluate FED string
Figure 2–42	?FRDIS	Disassemble an instruction
Figure 2–43	?FRITS	Insert a temporary symbol
Figure 2–44	?FRDTS	Delete a temporary symbol

The operating system returns information to the second offset for all function codes except ?FRCR and ?FROST.

Table 2–24 describes the contents of the packet to evaluate a FED string.

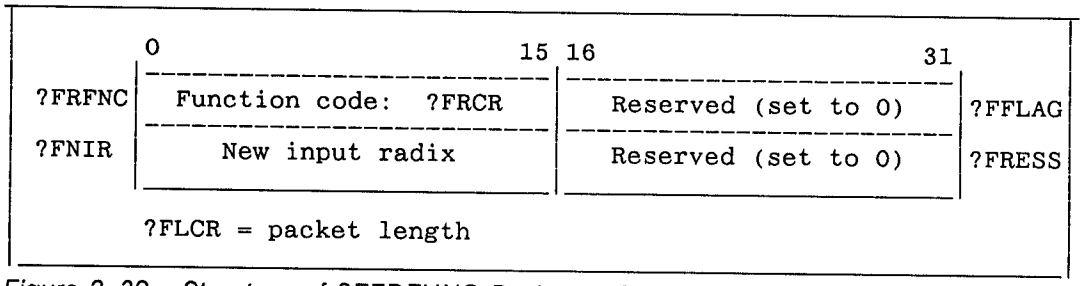


Figure 2–39. Structure of ?FEDFUNC Packet to Change Radix

Licensed Material - Property of Data General Corporation

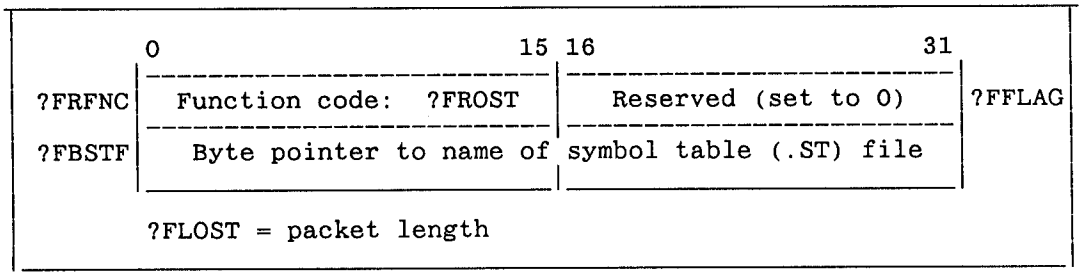


Figure 2–40. Structure of ?FEDFUNC Packet to Open Symbol Table File

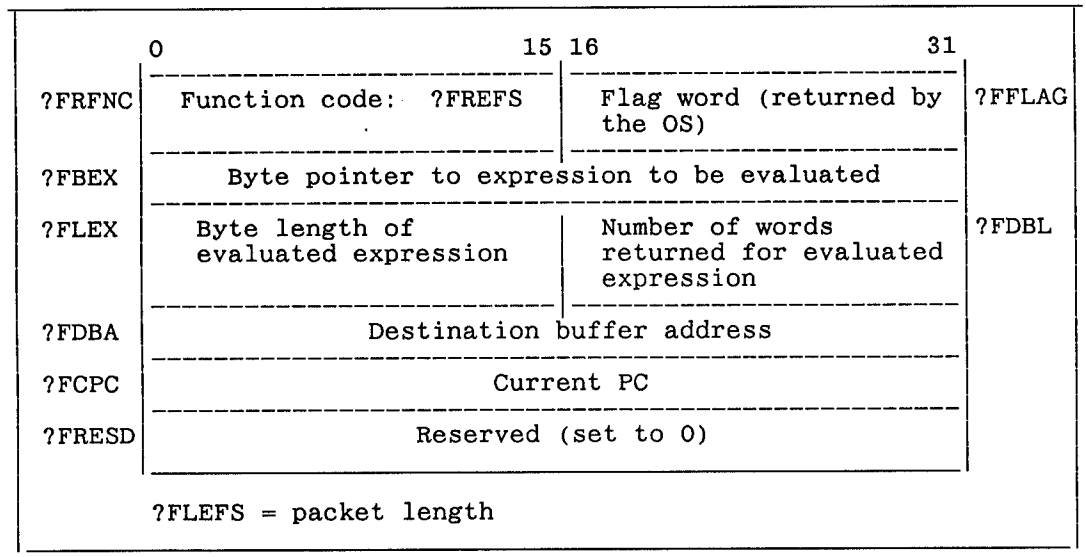


Figure 2–41. Structure of ?FEDFUNC Packet to Evaluate a FED String

?FEDFUNC Continued

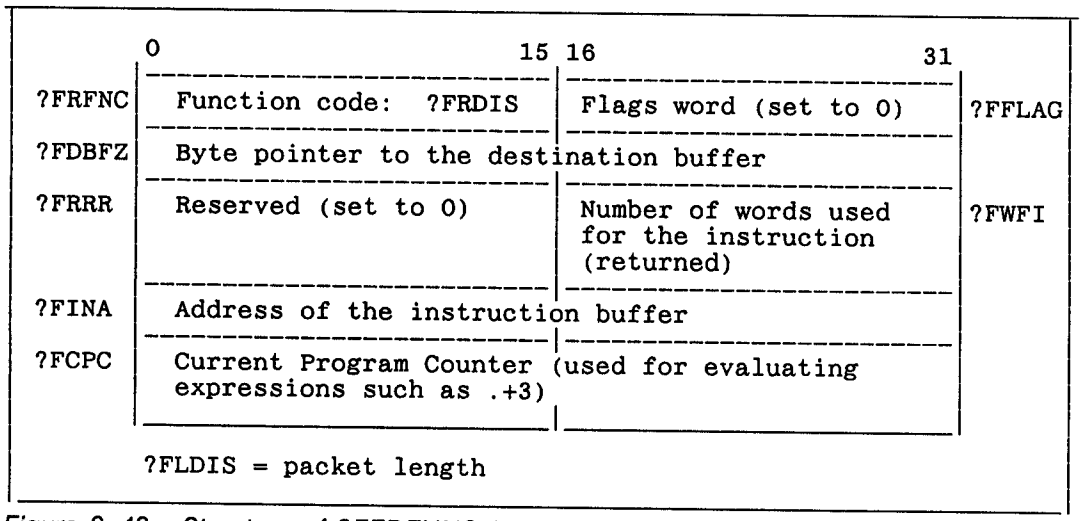


Figure 2–42. Structure of ?FEDFUNC Packet to Disassemble an Instruction

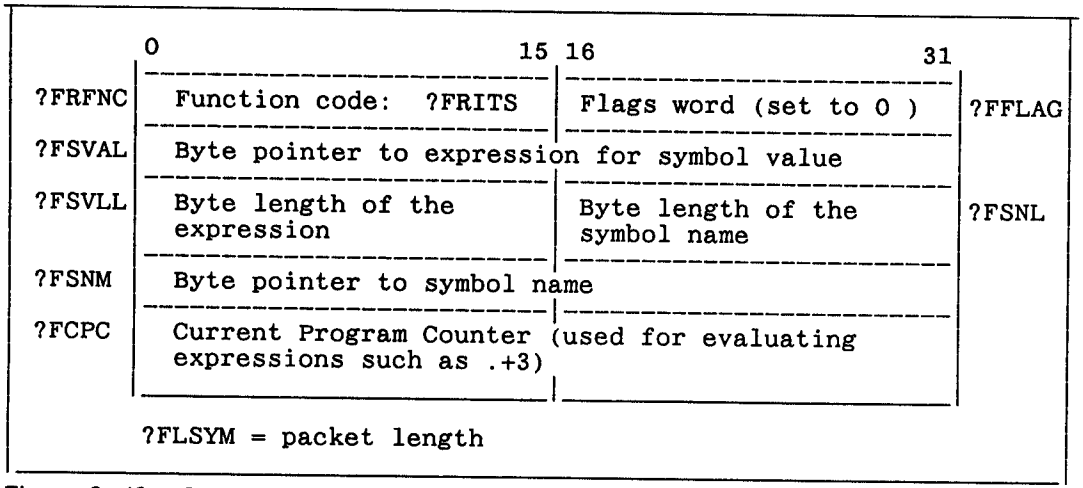


Figure 2–43. Structure of ?FEDFUNC Packet to Insert a Temporary Symbol

Licensed Material – Property of Data General Corporation

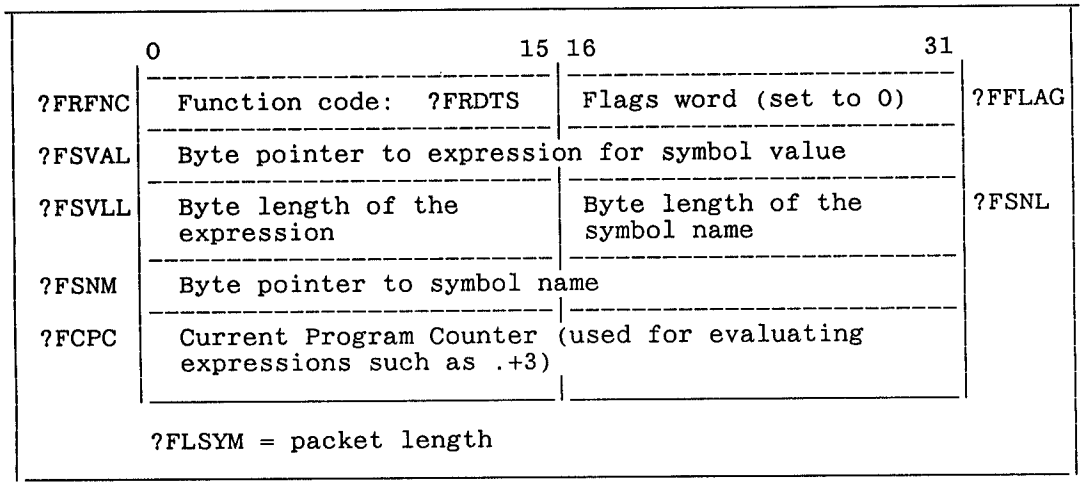


Figure 2-44. Structure of ?FEDFUNC Packet to Delete a Temporary Symbol

Table 2-24. Contents of ?FEDFUNC Packet to Evaluate a FED String*

Offset	Contents
?FRFNC	Contains a code that defines the function that you wish to perform. In this case, the function code is ?REFS.
?FFLAG	The OS can return the following flag words to this offset: - ?FINST, which means that the string contains an instruction. - ?FFLPT, which means that the string contains a floating-point number. - ?FEXPR, which means that the string contains an expression.
?FBEX (double word)	Byte pointer to the expression that you want to evaluate.
?FLEX	The length of the expression that you want to evaluate (in bytes).
?FDBL	The number of flag words that the OS has returned in offset ?FFLAG for the expression that you want to evaluate.
?FDBA (double word)	Address of the destination buffer. (The destination buffer that you provide must be at least ?FMEFS bytes long.)
?FCPC (double word)	The current PC. (?FEDFUNC uses the current PC to evaluate expressions such as .+3.)
?FRESD (double word)	Reserved. (Set to 0.)

* There is no default unless otherwise specified.

?FEOV

Forces end-of-volume on labeled magnetic tape.

?FEOV
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Number of channel on which the magnetic tape was opened
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

EREOF	End of file
ERIFT	Illegal file type; file is not a labeled tape

File System Error Codes

Why Use It?

?FEOV allows you to end processing on the current labeled tape volume and swap to the beginning of the next volume.

What It Does

For output tapes, ?FEOV causes the operating system to flush the current buffer to tape and process the end-of-volume normally. Then, a swap to the next reel occurs.

For input tapes, ?FEOV causes the operating system to space the tape to its logical end. If the operating system detects end-of-volume labels, it swaps to the next volume. However, if the operating system detects end-of-file labels, it returns the error code EREOF in AC0.

Licensed Material - Property of Data General Corporation

?FLOCK

Locks a file element.

?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	Element already locked
ERLNG	Lock not granted — pended request canceled
ERLLE	Lock limit exceeded — more than 1023 locks for one element
ERFNO	Channel not open
ERICN	Illegal channel number
ERVWP	Invalid address passed as system call argument

Why Use It?

Use this system call to restrict access to objects. You can use it to prevent cooperating processes from accessing the same object at the same time. With shared locks several processes may use an object simultaneously (e.g., reading records), but no one may gain exclusive access (perhaps for writing).

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 of the particular element in offset ?FLSEL of the parameter packet.

You must have issued ?OPEN for a file before you can lock one of its elements. Place the channel number ?OPEN returns in the ?FLOCK packet along with the type of lock.

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

Licensed Material - Property of Data General Corporation

?FLOCK Continued

Figure 2–45 shows the structure of ?FLOCK's parameter packet, and Table 2–25 describes its contents.

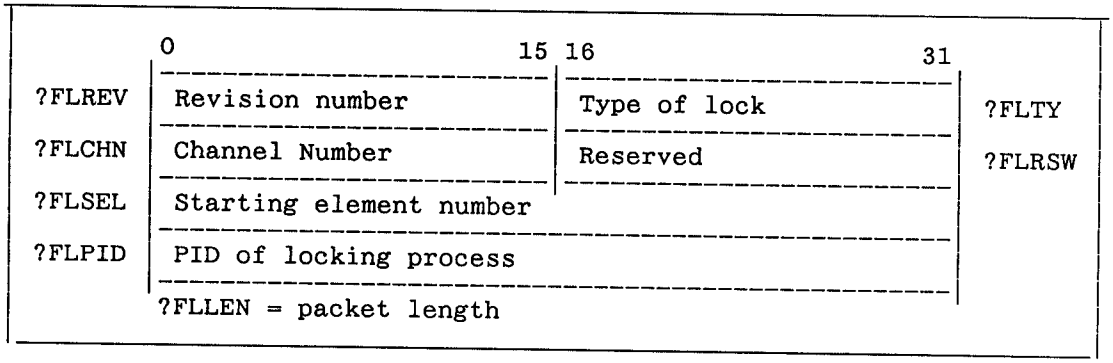


Figure 2–45. Structure of ?FLOCK Packet

Table 2–25. Contents of ?FLOCK Packet*

Offset	Contents
?FLREV	Packet revision number. Place zero here.
?FLTY	Type of lock. It specifies the operation you want. Select from the following values. ?FTCK -- Check for a lock. If selected, then AOS/VS does no locking but it returns the status of the element (or range) specified. If it is unlocked, then ?FLTY contains 0; if it is locked, then ?FLTY contains the lock type (?FTSH or ?FTEX). ?FLTY might return ?FWFL instead. ?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. ?FTEX -- Create an exclusive lock. ?FTSH -- Create a shared lock. ?FWFL -- Lock the whole file.
?FLCHN	Supply the channel number for the file.
?FLRSW	Reserved. (Set to 0.)
?FLSEL (double word)	Starting element number. Supply this number to identify the element to lock or the first element of a range of elements to lock. Your process and cooperating processes 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

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.

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.

Licensed Material – Property of Data General Corporation

?FSTAT

Gets file status information.

?FSTAT *[file status packet address]*
error return
normal return

Input

AC0 may contain one of the following:

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

AC1 Flags:

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

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.)

Licensed Material - Property of Data General Corporation

What It Does

?FSTAT returns the target entry's status parameters to the file status packet you define.

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 you set Bit 0 of AC1, the operating system resolves the link entry and returns status information about the file to which the link refers. If the pathname ends in a link and Bit 0 of AC1 equals 0, the operating system returns status information about the link without resolving it.

The format of the ?FSTAT information varies, depending on the file type of the target 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 2–46, Figure 2–47, Figure 2–48, and Figure 2–49 show the structure of the various ?FSTAT packets.

*

?FSTAT Continued

Unit File/IPC File Packets

In the packet for unit files, offset ?SDCU contains the unit's device code and unit number. (See Figure 2–46.) The comparable offset in the ?FSTAT IPC packet is ?SPNH. (See Figure 2–47.) Offset ?SPNH contains the IPC file's global port number. Offset ?STCH specifies the time the file was created. This parameter contains the same information as offset ?TCTH in the ?CREATE system call's time block.

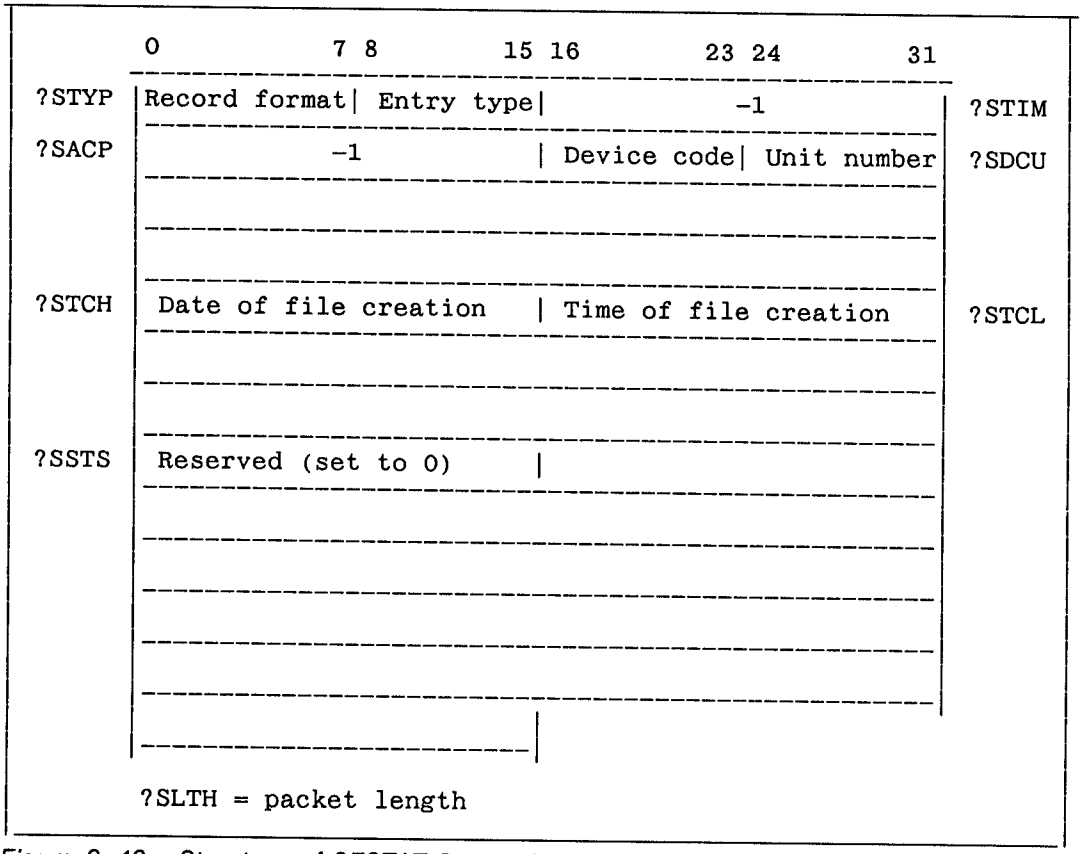


Figure 2–46. Structure of ?FSTAT Packet for Unit Files

Licensed Material - Property of Data General Corporation

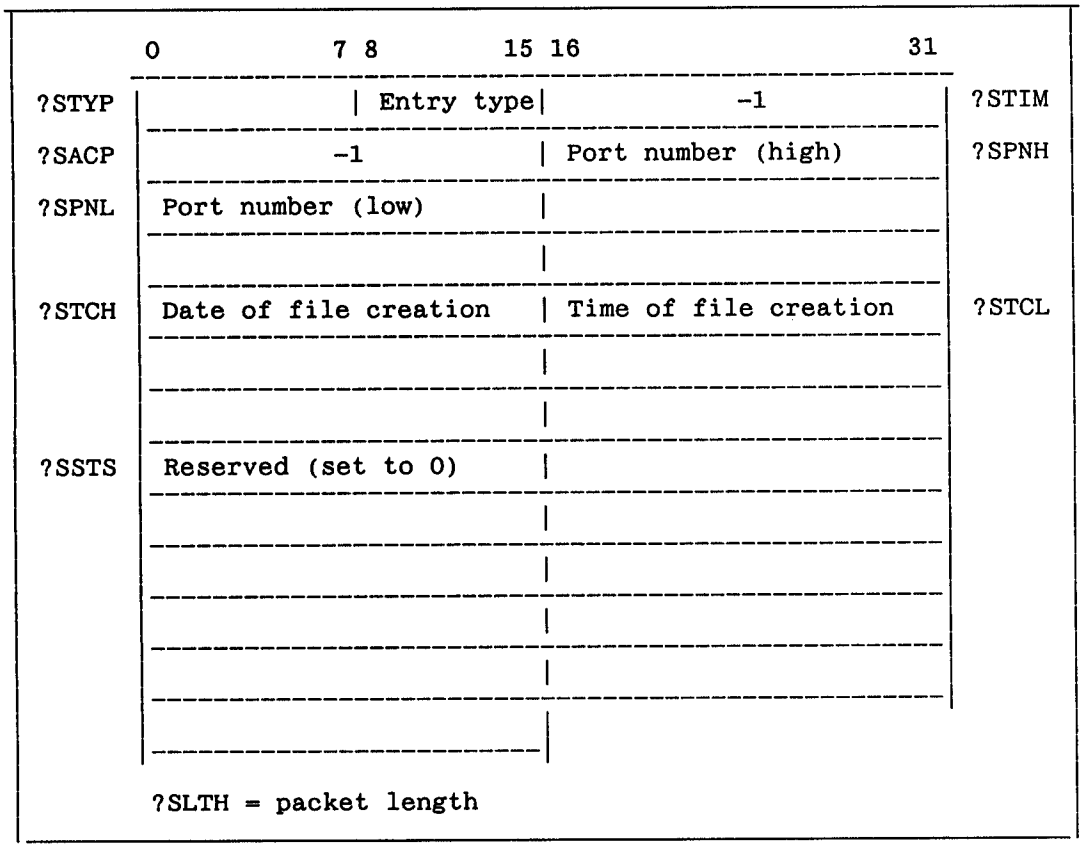


Figure 2–47. Structure of ?FSTAT Packet for IPC Files

?FSTAT Continued

Directory/Remaining Types Packets

If the target entry is a directory or a control point directory, set the right byte of offset ?SHFS to the directory's hash frame size. (See Figure 2-48.)

The operating system supplies two parameters for control point directories and logical disks only. Offset ?SMSH records the maximum amount of disk space available in the directory or logical disk (that is, the MS value). Offset ?SCSH indicates the directory's or the logical disk's current size in blocks (that is, its CS value). Note that these values apply only if the target entry is file type ?FCPD (control point directory) or file type ?FLDU (logical disk).

	0 7 8 15 16 31		
?STYP	Entry type	-1	?STIM
?SACP	-1	Hash frame size	?SHFS
?SLAU	Reserved (set to 0)	CPD maximum size (high)	?SMSH
?SMSL	CPD maximum size (low)	Max. number index levels	?SMIL
?STCH	Date of file creation	Time of file creation	?STCL
?STAH	Date of last file access	Time of last file access	?STAL
?STMH	Date of last file modification	Time of last file modification	?STML
?SSTS	Reserved (set to 0) (See Table 2-26 and Figure 2-50.)	Reserved (set to 0)	?SEFW
?SEFH	Reserved (set to 0)	File size [bytes] (high)	?SEFM
?SEFL	File size [bytes] (low)	Starting LDU addr (high)	?SFAH
?SFAL	Starting LDU addr (low)	Current No. index levels	?SIDX
?SOPN	Open count	CPD current size (high)	?SCSH
?SCSL	CPD current size (low)		

?SLTH = packet length

Figure 2-48. Structure of ?FSTAT Packet for Directory Files

Offset ?SMIL indicates the maximum number of index levels allowed for both standard and control point directories.

In the packet for the other file types, offset ?SDEH specifies the target file's element size, offset ?SMIL specifies the maximum index level, and ?SCPS specifies the file control parameters. (See Figure 2-49.) Offset ?SCPS, which applies only to files with fixed-length records, equals the file's maximum record length. Offset ?SCPS equals the Host ID for network type files. The contents of ?SCPS are undefined for other record types.

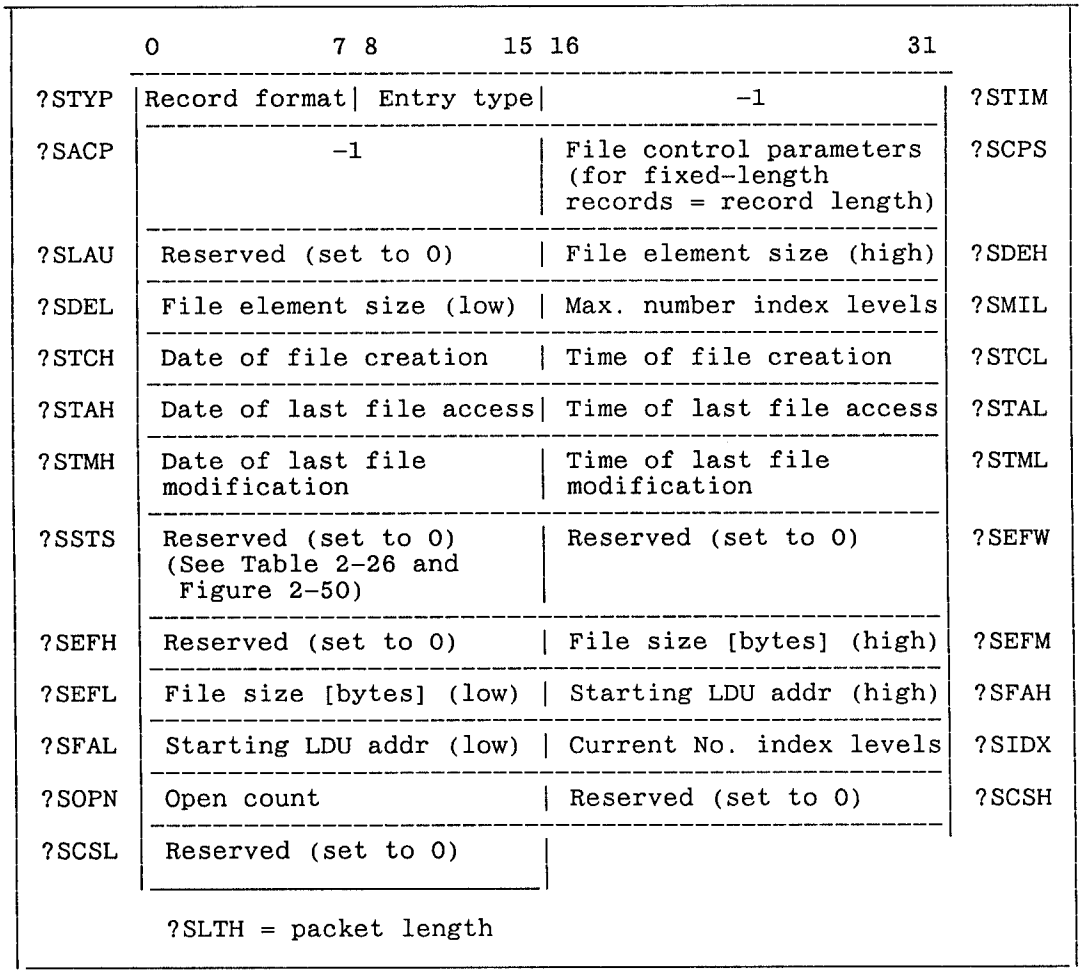


Figure 2–49. Structure of ?FSTAT Packet for Other File Types

Offsets ?STCH through ?STML specify the time the file was created, the time it was last accessed, and the time it was last modified. These offsets correspond to the parameters in the ?CREATE system call's time block.

The operating system describes certain characteristics of the entry by returning one or more flag bits to ?SSTS, the file status word. See Table 2-26 for a complete list of these flags.

Table 2-26. Flags Returned in Offset ?SSTS

Flag	Meaning
?FSHB	The file is a shared file.
?FMDB	The file was modified.
?FPRM	The file or directory has the Permanence attribute.
?FDLE	The O/S deletes the file after the last ?CLOSE.
?FUDA	The file has an associated UDA (user data area).
?FARA	All users have read access to the file or directory.
?FAEA	All users have execute access to the file or directory.

Figure 2-50 shows the structure of offset ?SSTS.

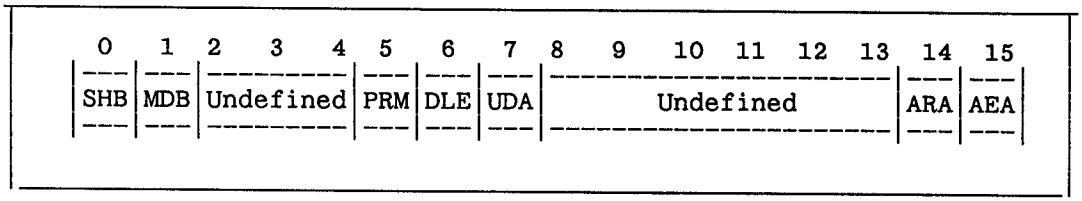


Figure 2-50. ?SSTS Structure

Offset ?SEFM specifies the file's byte length.

Offset ?SFAH specifies the file's starting address in the logical disk. This is a double-precision integer, and can range from 0 to the highest block on the logical disk minus 1.

Offset ?SIDX specifies the current number of index levels (indexes) for the file. This value should not exceed the value of ?SMIL, the maximum number of index levels allowed to the file.

The open count, offset ?SOPN, records the number of users currently using the file for I/O (the number of ?OPEN calls without comparable ?CLOSE calls).

You can only retrieve the length of a file via ?FSTAT after you have opened, reopened, or just created the file. In other words, you cannot retrieve the length correctly while you are currently writing to the file.

Sample Packet

The following is a sample of an ?FSTAT IPC file packet:

```
PKT:    .BLK    ?SLTH           ;Allocate enough space for the packet
                                ;(packet length = ?SLTH).
        .LOC    PKT+?STYP       ;Entry type.
        .WORD   0               ;Entry type for IPC files returned.
        .LOC    PKT+?SPNH       ;Port number.
        .DWORD  0               ;Port number returned.
        .LOC    PKT+?STCH       ;File creation date.
        .WORD   0               ;File creation date returned.
        .LOC    PKT+?STCL       ;File creation time.
        .WORD   0               ;File creation time returned.
        .LOC    PKT+?SSTS       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?SLTH       ;End of packet.
```

Notes

- See the description of ?RECREATE in this chapter.

?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.

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.

Licensed Material - Property of Data General Corporation

?FUNLOCK — Unlocks a file element.

?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

Why Use It?

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

What It Does

?FUNLOCK releases a lock that your process or another 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 elements. 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 element numbers so they can guarantee successful unlocking.

?FUNLOCK Continued

Figure 2-51 shows the structure of ?FUNLOCK's parameter packet, and Table 2-27 describes its contents.

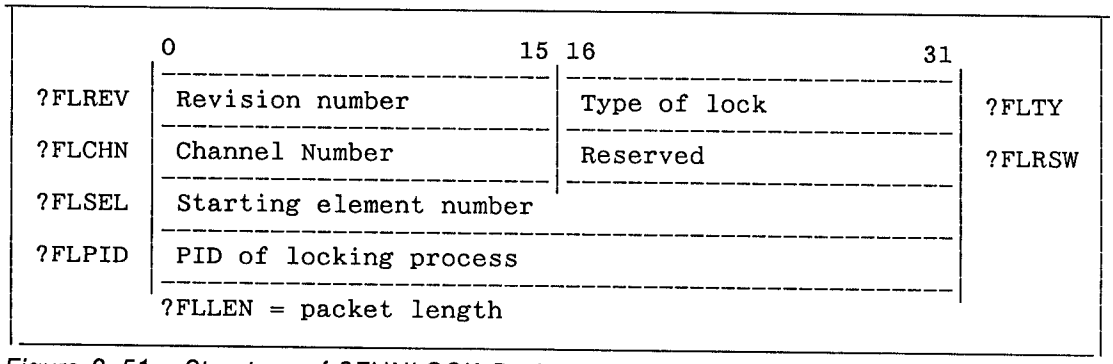


Figure 2-51. Structure of ?FUNLOCK Packet

Table 2-27. Contents of ?FUNLOCK Packet*

Offset	Contents
?FLREV	Packet revision number. Place zero here.
?FLTY	Type of lock. 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.
?FLCHN	Supply the channel number for the file.
?FLRSW	Reserved. (Set to 0.)
?FLSEL (double word)	Starting element number. Supply this number to identify the element to unlock or the first element of a range of elements to unlock. 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

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

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.

?GACL Continued

What It Does

?GACL returns the access control list (ACL) associated with a file or directory.

To use ?GACL, the calling process must have Read and Execute access to the target entry's parent directory, or — if you have Owner access to the file — Execute access to the parent directory. Before you issue ?GACL, set up a receive buffer for the ACL. The symbol ?MXACL represents the maximum byte length of the buffer. The target file or directory can be specified in one of two ways: either by a byte pointer to the target entry, or by using offset ?GCPCN in the ?GACL packet. The use of a packet is only necessary if you choose to specify the target entry's channel number. AC0 must contain zero to indicate the use of a packet. Load AC1 with a byte pointer to the receive buffer before you issue ?GACL.

The operating system returns the ACL to the receive buffer in the format used for ?DACL and ?SACL. This format is

username<0>accesstype[...]<0>

For example, if the CLI command

ACL FOO BRIAN,OAR MIKE,RE

has executed, then issuing ?GACL to file FOO results in the receive buffer's containing the following 14 bytes in its leftmost bytes.

BRIAN<0><?FACO+?FACA+?FACR>MIKE<0><?FACR+?FACE><0>

Each of symbols <?FACO+?FACA+?FACR> and <?FACR+?FACE> requires 1 byte.

Figure 2-52 shows the structure of the ?GACL packet.

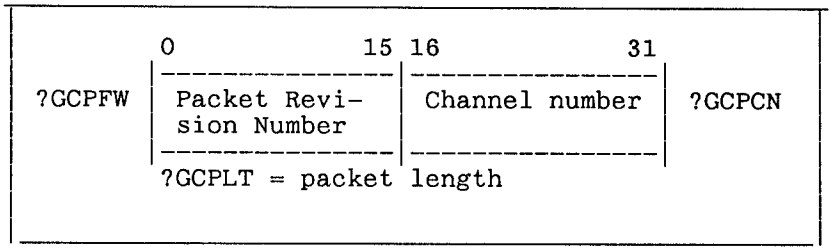


Figure 2-52. Structure of ?GACL Packet

Notes

- See the descriptions of ?DACL and ?SACL in this chapter.

?GBIAS — Gets the current bias factor values.

?GBIAS
error return
normal return

Input

None

Output

AC0 Contains the following:

- Bits 16 through 23 contain the maximum bias factor
- Bits 24 through 31 contain the minimum bias factor

AC1 Undefined

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?GBIAS lets you determine your system's maximum and minimum bias factors at runtime. You can use ?GBIAS in conjunction with ?SBIAS, which sets the bias factor values.

What It Does

?GBIAS returns the maximum and minimum bias factors for your system to the low-order bits of AC0. If there is no maximum bias factor, the operating system returns 0 to AC0. The default minimum bias factor is zero. ?SBIAS can change it.

Notes

- See the description of ?SBIAS in this chapter.
- Refer to your operating system's "How to Generate and Run" manual for information on the system-generation procedure and the criteria for setting the bias factors.

?GCHR

Reads device characteristics of a character device.

?GCHR
error return
normal return

Input

AC0	One of the following: • Byte pointer to the name of the target device • Channel (number) on which the device is open	
AC1	Flag bits: Bit 0 = 0 if AC0 contains a byte pointer Bit 0 = 1 if AC0 contains a channel number Bit 1 = 0 to get the current characteristics of the device Bit 1 = 1 to get the default characteristics of the device	
AC2	Address of a 3-word area designated to receive the device characteristics (See Table 2-28.)	

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERICN	Illegal channel number
ERIFT	Illegal file type
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?GCHR can be a useful preliminary call for ?SCHR, which sets a character device's characteristics. (Note that you can also set what are known as "extended characteristics" with ?SECHR and use ?GECHR to obtain them.)

Licensed Material - Property of Data General Corporation

What It Does

?GCHR returns the default or the current characteristics of a character device, depending on the value you place in Bit 1 of AC1. A character device's characteristics (distinguishing features) supply descriptive information that the operating system uses to tailor the device's I/O. The operating system returns the characteristics of the target device to the 3-word buffer you specify in AC2. (See Table 2-28.)

Table 2-28. Character Device Characteristics Words

Word	Characteristic	Meaning
0	?CST	Simulates 8-column tabs.
	?CSFF	Simulates form feeds.
	?CEPI	Requires even parity on input.
	?C8BT	Allows 8 data bits per character.
	?CSPO	Sets even parity on output.
	?CRAF	Sends 17. rubout characters after each form feed, unless the form feed is simulated.
	?CRAT	Sends two rubout characters after each tab, unless the tabs are simulated.
	?CRAC	Sends two rubout characters after each carriage return or NEW LINE.
	?CNAS	Non-ANSI-standard device.
	?COTT	Converts octal 175 and octal 176 to octal 33 (ESC) on output.
	?CEOL	No automatic carriage return or line feed at end of each line. (If ?CWRP is off and ?CEOL is on, then the output truncates at the line length defined by the device characteristics.)
	?CUCO	Uppercase output only.
	?CMRI	Monitors ring indicator on modem control line.

(Continues)

Table 2-28. Character Device Characteristics Words

Word	Characteristic	Meaning
0 (Cont.)	?CFF	Form feed on open.
	?CEB0+?CEB1	Echo mode (one of the following): EB0 | EB1 || Effect 0 | 0 || No echoing 0 | 1 || ?CEOS; echo exactly as input 1 | 0 || ?CEOC; default -- echo control characters with uparrows (for example, ^A) 1 | 1 || Undefined
1	?CULC	Both lowercase and uppercase terminal.
	?CPM	Page mode.
	?CNRM	Disables message receipt (accept no ?SEND system calls).
	?CMOD	Modem control. (See "Full-Duplex Modems".)
	?CDT0+?CDT3	Device type (one of the following): ?TTY -- 4010A hard-copy console. ?CRT3 -- 6052 or 6053 display console. ?CRT6 -- 6130 display console.
	?CTO	Enables device time-outs.
	?CTSP	Retains trailing blanks (card readers only).
	?CPBN	Right-justifies each column to 16-bits (card readers only).
	?CESC	Each ESC character generates a CTRL-C CTRL-A interrupt.
	?CWRP	(Hardware) Wraparound for lines longer than the characters per line defined in the last characteristic word.
	?CFKT	Uses function keys as input delimiters (device types ?CRT3 and ?CRT6 only).
	?CNNL	Does not append NEW LINE characters.

(Continued)

Table 2-28. Character Device Characteristics Words

Word	Characteristic	Meaning
2	Left byte	Number of lines per page (for Page mode devices).
	Right byte	Characters per line (CPL).

(Concluded)

Table 2-29 lists several of the more common device characteristics and their meanings.

Table 2-29. Commonly Used Device Characteristics

Characteristic	Meaning If Chosen
?CNAS	The device is a non-ANSI-standard console. (See "ANSI-Standard Versus Non-ANSI-Standard Consoles," for descriptions of the two types.)
?COTT	The device converts input values of octal 175 and octal 176 to octal 33 (ESC) when it receives them.
?CEB0 ?CEB1	Echoing mode--the following bit values define the echo mode variations: 0 no echoing. ?CEOS echoes the characters exactly as input. ?CEOC echoes control characters with uparrows (for example, CTRL-C becomes ^C), escape characters echo as dollar signs ($).
?CULC	Retains uppercase and lowercase characters, rather than converting lowercase characters to their uppercase equivalents.

(Continues)

Table 2-29. Commonly Used Device Characteristics

Characteristic	Meaning If Chosen
?CPM	Device is in Page mode. The OS stops the output automatically when the output reaches the number of lines per page you specified or a form-feed character. Form feed frees the output until you release it with CTRL-Q. (Omit ?CPM if you want form-feed characters to pass normally.)
?CMOD	Modem control. ?CMOD must be set during the system-generation procedure.
?CNNL	Ignores NEW LINE characters. ?CNNL applies to card readers only.
?CDT0 ?CDT1 ?CDT2 ?CDT3	Device type. The OS defines a device's type based on the device's handling of the ESC character and certain control characters. The following bit masks define the device types: ?TTY -- 4010A hard-copy console. ?CRT3 -- DASHER display console. ?CRT6 -- D400/D450 display console.
?C8BT	For 8-bit ASCII consoles only. ?C8BT passes each byte between an Asynchronous Line Multiplexor (ALM) and the user buffer without modification. This characteristic is illegal for DASHER consoles and for other consoles that use the high-order bit (of the byte) for parity.
?CTSP	For card readers only. ?CTSP retains trailing blanks.
?CPBN	For card readers only. ?CPBN right-justifies each card column into a 16-bit memory word.
?CMRI	Monitor Ring Indicator. ?CMRI must be set during the system-generation procedure.

(Concluded)

ANSI-Standard Versus Non-ANSI-Standard Consoles

Bit ?CNAS in the first device characteristics word defines your console as ANSI standard or non-ANSI standard. The difference between ANSI-standard and non-ANSI-standard consoles is in how the operating system treats the carriage return (CR), line feed (LF), and NEW LINE keys.

When you type a NEW LINE character on an ANSI-standard console, the operating system echoes the NEW LINE character and the terminal performs a line feed and a carriage return.

On non-ANSI-standard consoles, the operating system translates each carriage return to a NEW LINE and each line feed to a carriage return. When the operating system outputs a line feed to a non-ANSI-standard console, it precedes the line feed with a carriage return.

Notes

- See the descriptions of ?SCHR, ?SECHR, and ?GECHR in this chapter.
- Refer to PARU.32.SR for the assembled values of the character device characteristics.

?GCLOSE Closes a file previously opened for block I/O.

?GCLOSE
error return
normal return

Input

AC0 Reserved (Set to 0.)

AC1 Contains the following:

- Bit 0 is a flag bit:

  Bit 0 = 1 to modify status on output

  Bit 0 = 0 otherwise

- Bits 1 through 31 contain the channel number associated with the target file

AC2 Reserved (Set to 0.)

Output

AC0 Undefined

AC1 Contains the following:

- Bit 0, if set to 1 on input, contains one of the following on output:

  0 if the file was not modified
  1 if the file was modified

- Bits 1 through 31 are unchanged

AC2 Undefined

Error Codes in AC0

ERACU	Attempt to close unopen channel/device
ERCIU	Channel in use
ERSIM	Simultaneous requests on same channel

Why Use It?

You must use ?GCLOSE, rather than the standard ?CLOSE, to close a file that was previously opened for block I/O. Like ?CLOSE, ?GCLOSE breaks the target file's channel assignment, which allows the operating system to reassign that channel number to another file.

Licensed Material - Property of Data General Corporation

What It Does

?GCLOSE closes a file previously opened with ?GOPEN for block I/O operations. If you set bit 0 of AC1 on input, the operating system returns the file's status (modified or not modified) to AC1 on output.

If you are performing block I/O on a magnetic tape, and you issue ?GCLOSE, the operating system follows the last block written on the tape file with a logical end-of-tape mark (two consecutive tape marks).

Don't use ?GLOSE to close a file you've opened with ?OPEN. Use ?CLOSE instead.

Notes

- See the description of ?GOPEN in this chapter.

?GCPN

Gets the console port number.

?GCPN
error return
normal return

Input

AC0 One of the following:

- PID of the target process
- Byte pointer to the name of the target process
- −1 to get the calling process's console port number

AC1 One of the following:

- 0 if AC0 contains a PID
- −1 if AC0 contains a byte pointer

Otherwise, the O/S ignores AC1

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Global port number of the target process's console

AC2 Undefined

Error Codes in AC0

ERFDE	File does not exist
ERPOR	PID is out of range for this process
ERPRH	Attempt to access process not in hierarchy
	Parameter address error

Licensed Material - Property of Data General Corporation

Why Use It?

?GCPN gets the global port number of a console for you, even if you know only the name or PID of its associated process.

What It Does

?GCPN returns the global port number of the target process's console.

Before you issue ?GCPN, load AC0 with the PID of the target process, a byte pointer to the name of the target process, or −1. If you specify −1, the operating system returns the port number of the console that is associated with the calling process. If the operating system cannot find the PID or the name that you specified, or if there is no console associated with the target process, then it returns either error code ERFDE or error code ERPRH to AC0.

?GCRB

Gets the base of the current resource. (16–Bit Processes Only)

?GCRB
normal return

Input

None

Output

AC0	Base address of the current resource
AC1	Undefined
AC2	Undefined

Error Codes in AC0

No error codes are currently defined. If an error occurs, the operating system exits via ?BOMB. The state of AC0 is undefined.

Why Use It?

?GCRB is useful for finding out the logical address of an element in any overlay bound for a multiple overlay area. (?GCRB assumes that you know the element's displacement relative to the overlay area base.)

What It Does

?GCRB returns a label to AC0 that indicates the base address of the current resource. You can use this information with the resource system calls to determine the current base–relative offset of a movable resource. (A movable resource is an overlay that the operating system can relocate to any basic area within a multiple overlay area.) To determine the absolute address of an offset in a particular resource, add the relative value of that offset to the value of the resource base.

Note that ?GCRB has no error return. If the operating system encounters an invalid resource base, it transfers control to the ?BOMB error–detection routine.

Licensed Material – Property of Data General Corporation

?GDAY

Gets the current date.

?GDAY
error return
normal return

Input

None

Output

AC0	Day from 1 through 31
AC1	Month from 1 through 12
AC2	Year minus 1900

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?GDAY to write the date on output page headers.

What It Does

?GDAY gets the current date from the system clock, and returns it as day, month, and year to AC0, AC1, and AC2, respectively. Notice that the value for the current year is an offset from a base of 1900.

Licensed Material - Property of Data General Corporation

?GDLM

Gets a delimiter table.

?GDLM
error return
normal return

Input

AC0 One of the following:

- byte pointer to the device name
- channel number of the file

AC1 Flag bits:

- ?SDCN if AC0 contains a channel number
- ?SDDN if AC0 contains a byte pointer to a device name

AC2 Word address of the 16-word area to receive the delimiter table.

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERFDE	File does not exist
ERICN	Illegal channel number
ERIDP	Illegal destination port
ERIFT	Illegal file type
ERMPR	System call parameter address error (the receive buffer is not in the unshared area of your address space)
ERPRV	Caller not privileged for this action
ERVBP	Invalid byte pointer
ERVWP	Invalid word pointer

Licensed Material - Property of Data General Corporation

Why Use It?

?GDLM is useful whenever you want to save the current delimiter table. For example, you may need to set up a temporary delimiter table for a special situation, but later revert to the original delimiter table.

What It Does

?GDLM returns a delimiter table for a file with data-sensitive records or for a character device. You can set a delimiter table either with ?SDLM or ?OPEN.

Before you issue ?GDLM, reserve a table of 16 consecutive words in your address space and load the word address of the table into AC2. (For information on the structure of delimiter tables, see the description of the ?OPEN system call.)

Notes

- See the descriptions of ?SDLM and ?OPEN in this chapter.

?GECHR — Get extended characteristics.

?GECHR
error return
normal return

Input

AC0 contains one of the following:

- Byte pointer to the name of the target device
- Channel (number) on which the device is open

AC1 contains the following flags:

bit 0 = 0	If AC0 contains a byte pointer
bit 0 = 1	If AC0 contains a channel number
bit 1 = 0	To get the current characteristics
bit 1 = 1	To get the default characteristics
bits 28 – 31	Length of the packet in words, between values ?CLMIN and ?CLMAX inclusive

AC2 contains the address of a ?CLMIN to ?CLMAX word area to receive the device characteristics.

Output

AC0 Unchanged

AC1 Number of characteristic words returned

AC2 Unchanged

Licensed Material – Property of Data General Corporation

Error Codes in AC0

ERICN	Illegal channel
ERIFT	Illegal file type
ERPRE	Invalid system call parameter
ERARG	Too few or too many arguments to PMGR. You attempted to define a packet with length less than ?CLMIN.
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

The ?GECHR call can be a useful preliminary call for ?SECHR which sets a character device's characteristics. Suppose an attempt is made to get characteristics on a device where the characteristics have no meaning, (e.g., the fourth characteristic word is only valid on a system with intelligent controllers that can deal with Baud rates in real time — MV/8000-II, MV/6000, MV/10000, MV/2000). Then, a bit (?CVAL) will be set to 0 in the fourth word to indicate the word is invalid in the current configuration. This call may also be used as a replacement for the ?GCHR call since the number of characteristic words is passed as an argument to the call.

What It Does

The ?GECHR call returns the current or default characteristics for a character device depending on the value placed in bit 1 of AC1. Since the length of the characteristic word packet is passed by the user in AC1, this call may be used to get any number of characteristic words up to ?CLMAX.

The system will return the number of characteristic words specified by AC1 to the buffer specified by AC2. Table 2-170, in the ?SCHR description later in this chapter, defines the character device characteristics currently supported. Refer to PARU.32 for the characteristic's assembled values.

The ?BMDEV characteristic indicates that the targeted device is a bit-mapped device, which implies that you can create windows on this device.

The ?CWIN characteristic indicates that the targeted device is actually a window on a physical bit-mapped device. You can use ?CWIN to determine whether you can issue ?GRAPHICS, ?PTRDEVICE, and ?WINDOW system calls.

The ?TRPE characteristic indicates that pointer events (e.g., mouse movements) delimit/terminate ?READ system calls. If you are interested in retrieving pointer events, then set this characteristic via ?SECHR.

Notes

- See the descriptions of ?GCHR, ?SCHR, and ?SECHR in this chapter.

?GHRZ — Gets the frequency of the system clock.

?GHRZ
error return
normal return

Input

None

Output

AC0 One of the following code values, which indicates the system clock's frequency:

Code	Frequency (Hz)
0	60
1	10
2	100
3	1000
4	50

AC1 Undefined

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

Your application may require you to know your system clock's real-time frequency. To get this information, you can use ?GHRZ.

What It Does

?GHRZ returns the system clock's real-time frequency, which was set during the system-generation procedure. The operating system returns the frequency as one of the code values listed above. (Real-time clock frequencies are measured in hertz (Hz).)

Notes

- Refer to your operating system's "How to Generate and Run" manual for information on setting the system clock's real-time frequency.

?GLINK — Gets the contents of a link entry.

?GLINK
error return
normal return

Input

AC0	Byte pointer to the link entry's pathname
AC1	Byte pointer to a receive buffer for the resolved pathname
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERIFT	Illegal file type (The target entry is not a link entry.)
ERRAD	Read access denied
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

Because ?GLINK resolves the pathname that the link entry represents, you can use ?GLINK to decide whether to delete an existing link entry and/or create a new one.

What It Does

?GLINK returns the contents of the target link entry to the receive buffer that you specify in AC1. The calling process must have Read access to the link entry's parent directory.

Before you issue ?GLINK, set up a receive buffer in your logical address space. (The symbol ?MXPL represents the maximum byte length of the buffer.) Next, load the accumulators with the appropriate values. Then, issue ?GLINK. If the pathname that you specify in AC0 is not the pathname of a link entry, ?GLINK fails and the operating system returns error code ERIFT to AC0.

?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

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.

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.

Licensed Material - Property of Data General Corporation

?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.

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.

What It Does

?GNAME returns the complete pathname of the pathname fragment that you specify in AC0, starting with the root. If you load AC0 with the prefix =, the operating system returns the pathname of the current working directory.

The calling process must have access (of any type) to the files in the pathname fragment or have Read access to their parent directories. However, the operating system does not check for access privileges if the pathname fragment consists of only a prefix (for example, =).

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.

For ?GNAME to work properly, the file does not have to be open.

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

Why Use It?

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

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, load AC1 with the correct channel number, which returns in the ?OPEN packet. You can specify the packet address as an argument to ?GNFN, or you can load it into AC2 before you issue ?GNFN. Figure 2-53 shows the structure of the ?GNFN packet, and Table 2-30 describes its contents.

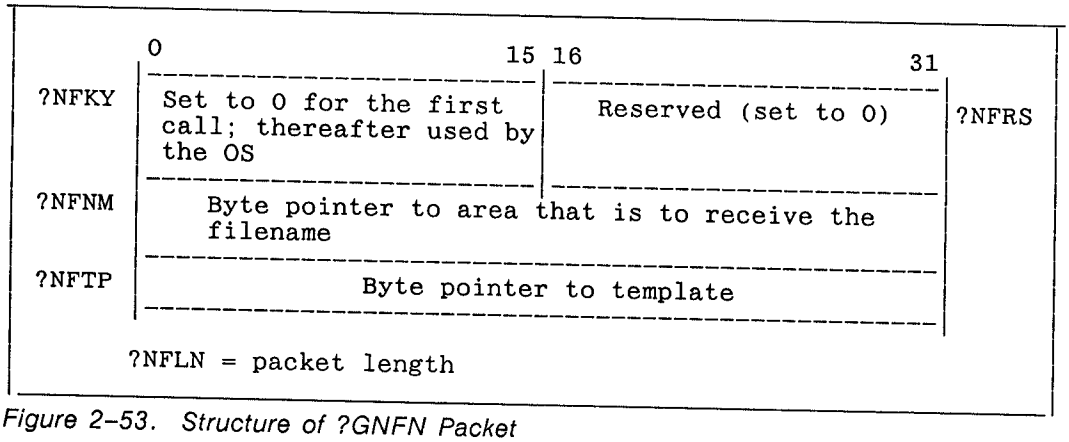


Figure 2-53. Structure of ?GNFN Packet

Table 2-30. Contents of ?GNFN Packet*

Offset	Contents
?NFKY	OS internal pointer. Set to 0 the first time time you issue ?GNFN; ignore thereafter.
?NFRS	Reserved. (Set to 0.)
?NFNM (double word)	Byte pointer to area that is to receive the filename.
?NFTP (double word)	Byte pointer to filename template, if present. DEFAULT = -1 (no template).

* There is no default unless otherwise specified.

Licensed Material - Property of Data General Corporation

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 ?NFNM. Therefore, you must use offset ?NFNM 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+?NFNM       ;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

Input

AC0 Byte pointer to the target file's filename

AC1 Target file's channel number, or -1, to direct the O/S 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

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

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.

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 2-54 and Table 2-31) and for all other file types (see Figure 2-55 and Table 2-32). 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.

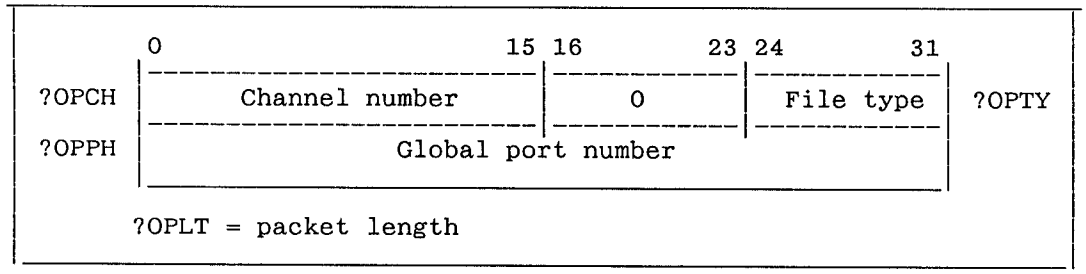


Figure 2-54. Structure of ?GOPEN Packet for IPC Files

Table 2-31. 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 (double word)	None.	Global port number.

* There is no default unless otherwise specified.

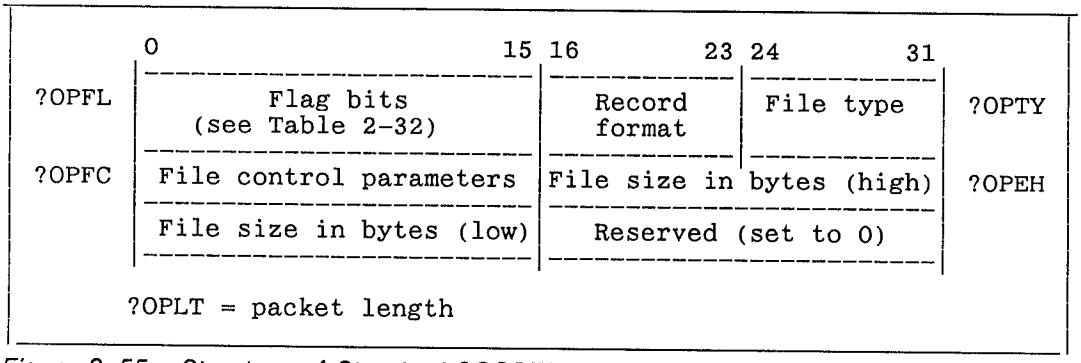


Figure 2-55. Structure of Standard ?GOPEN Packet

Table 2-32. 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 density mode, specify one of the following: ?OPDL--800 bpi. ?OPDM--1600 bpi. ?OPDH--6250 bpi. ?OPAM--automatic density mode selection.	Flag bit ?OPME--Exclusive open. Bits 1-15: Channel number.
?OPTY	None.	Left byte: Record format Right byte: File entry type
?OPFC	None.	File control parameters.
?OPEH (double word)	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.

?GOPEN Options

To open the file to the calling process exclusively, set bit ?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 setting one of the density-mode masks in offsets ?OPFL/?OPCH.

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

When you default the density-mode parameter (that is, omit the masks), 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).

?GOPEN Continued

?GOPEN Extension

The ?GOPEN extension is used to allow access to hardware options that exist for some Data General disk subsystems. These options (described below) are not 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 ?GOPEN extension is currently valid only when opening DPM or DPJ disk units. The extension information will be rejected for any other file type.

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 DPJ disks.
- Model option for some MTJ tapes (Model 6352 only).

To use the extension, you must set the extension bit (?OPXP) in the flag word (?OPFL) of the ?GOPEN packet. Figure 2-56 shows the structure of the ?GOPEN packet extension and Table 2-33 contains the option flags for offset ?ODF1 of the ?GOPEN packet extension.

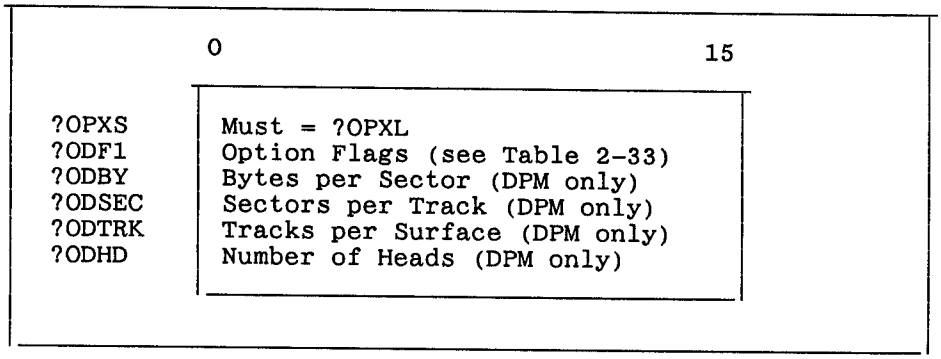


Figure 2-56. Structure of ?GOPEN Packet Extension

Table 2-33. 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)
?ODTP	Trespass if reserved by another port (DPJ only)
?ODST	Use streaming mode (MTJ Model 6352 only)

Table 2-34 contains the specific format selectors that Figure 2-56 introduces.

Table 2-34. 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+?OPCH       ;Channel number.
        .WORD   0               ;The OS returns this value.
        .LOC    PKT+?OPTY       ;File type and record format.
        .WORD   0               ;The OS returns this value.
        .LOC    PKT+?OPFC       ;Record length (if fixed format).
        .WORD   0               ;The OS returns this value.
        .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 your operating system's "How to Generate and Run" manual for information on the system-generation procedure.

?GPID Returns all active PIDs based on a host ID.

?GPID *[packet address]*
error return
normal return

Input

AC0	One of the following: • Host ID • −1 for the current host • Byte pointer to the host name
AC1	Defines contents of AC0 as follows: • 0 if AC0 contains a host ID • −1 if AC0 contains −1 • Ignored if AC0 contains a byte pointer
AC2	Address of the ?GPID packet, unless you specify the address as an argument to ?GPID

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?GPID packet

Error Codes in AC0

ERHID	Illegal host ID in AC0
ERHNE	Host does not exist
ERPRE	(AC1 = 0 and AC0 <> host ID) or (AC1 = 0 and AC0 <> −1) or (AC1 = −1 and AC0 doesn't contain a byte pointer)
ERPVS	Packet version not supported

?GPID Continued

Why Use It?

Use this system call to learn the PID numbers of all processes in use on a particular host. This knowledge is useful when you need to address all active processes on a host, such as when you implement functions similar to those of the ?.CLI macro and the Process Environment Display (PED) utility program.

What It Does

?GPID returns all (or at least as many as will fit in the buffer provided) PIDs that are active on the specified host.

Figure 2–57 shows the structure of ?GPID's parameter packet, and Table 2–35 describes its contents.

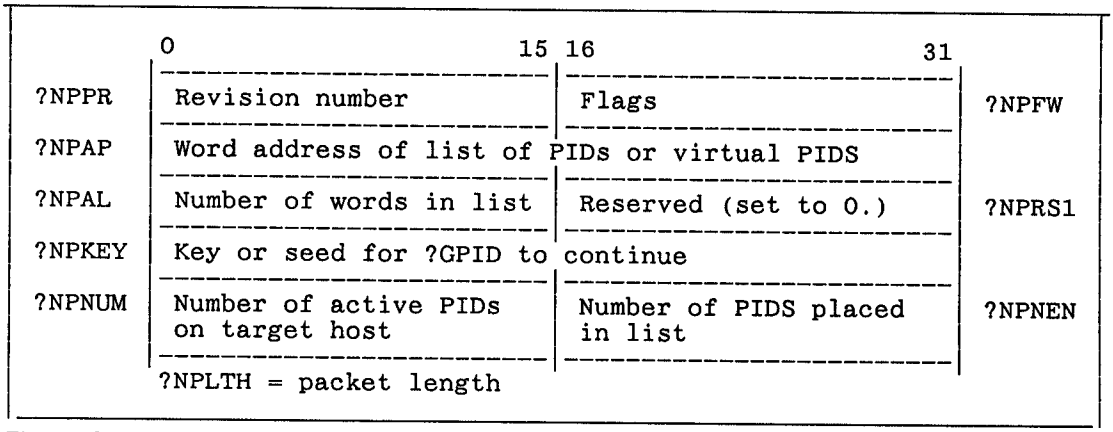


Figure 2–57. Structure of ?GPID Packet

Table 2-35. Contents of ?GPID Packet

Offset	Contents
?NPPR	Packet revision number. Place zero here.
?NPFW	Flags. Set all unused bits to zero. Since no bit offsets are currently defined, place zero here.
?NPAP (double word)	Word address of an array in which the OS will return the list of PIDs for active processes on the specified host. The list elements are either PIDs or virtual PIDs. Each entry in the array is one word wide.
?NPAL	Length in words of the array that receives the list of PIDs. ?GPID will not return more PIDs than the array will hold. The OS treats ?NPAL as an unsigned number.
?NPRS1	Reserved. (Set to 0.)
?NPKEY (double word)	Set to 0 the first time you issue this call. If you cannot allocate a large enough array to contain all the PIDs on the specified host, you must reissue the call to obtain the remaining PIDs. In this case, the prior issue of ?GPID has placed a number different from 0 in this offset; leave it here for all reissues. You know when you've obtained all the PIDs (and don't have to reissue ?GPID) when the rest of the buffer contains zeros or when ?NPNEN is less than the array length.
?NPNUM	The OS returns the number of PIDS that were active on the target host when ?GPID executed. You can use this number to adjust the size of the array. The O/S always returns ?NPNUM, even if ?NPAL contains zero.
?NPNEN	The OS returns the number of PIDs that it stored in the array. A program that uses this number doesn't have to count the number of unchanged entries to determine how many PIDs were returned. It's your responsibility to initialize the array. It's your responsibility to initialize the array. You can use the ?PIDS system call to find how many PIDs were generated and how many were active; then, using these values, determine the array size you want.

?GPORT

Returns the PID associated with a global port number.

?GPORT
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	32-bit global port number
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	PID or virtual PID associated with the global port number
AC2	Local port number that corresponds to the global port number in AC1

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?GPORT allows you to decode a global port number to its local equivalent and to obtain the PID (or virtual PID) of the port's owner. This information can be useful if you are sending or receiving IPC messages on multiple ports.

What It Does

?GPORT returns the PID (or virtual PID) and local port number associated with the global port number you specify in AC1. (Note that ?GPORT does not check the validity of the global port number you specify.)

Licensed Material - Property of Data General Corporation

?GPOS — Gets the current file–pointer position.

?GPOS *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the target file's I/O packet

Error Codes in AC0

ERICN	Illegal channel number
ERIFT	Illegal file type
ERFNO	Channel not open
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can use ?GPOS as a complement to ?SPOS, which repositions the file pointer, or as a complement to the pointer specification you set in the ?READ or ?WRITE packets. You can use a ?GPOS and ?SPOS sequence to perform conditional I/O on a file; for example, to check the position of the file pointer, and then reposition the pointer to skip over certain records in the file.

Using ?GPOS with the pointer specifications in the ?READ or ?WRITE packet gives you similar control over file I/O; that is, you might find the position of the file pointer with ?GPOS, and then adjust the ?READ or ?WRITE packets to read from the beginning of the file or write to the end of the file.

?GPOS Continued

What It Does

?GPOS returns the current position of the target file's file pointer to offset ?IRNH in the target file's current ?OPEN, ?READ, or ?WRITE packet. (See Figure 2-58.) The operating system expresses the file pointer position as a double-precision integer in this offset. Do not issue ?GPOS against a labeled magnetic tape file.

You can cite the address of the I/O packet as an argument to ?GPOS, or you can load the packet address into AC2 before you issue ?GPOS.

If the operating system returns 0 to offset ?IRNH and the target file is not a fixed-length file, the pointer is set to the first byte (character) in the file. Similarly, a 1 in ?IRNH means the file pointer is set to the second byte in the file.

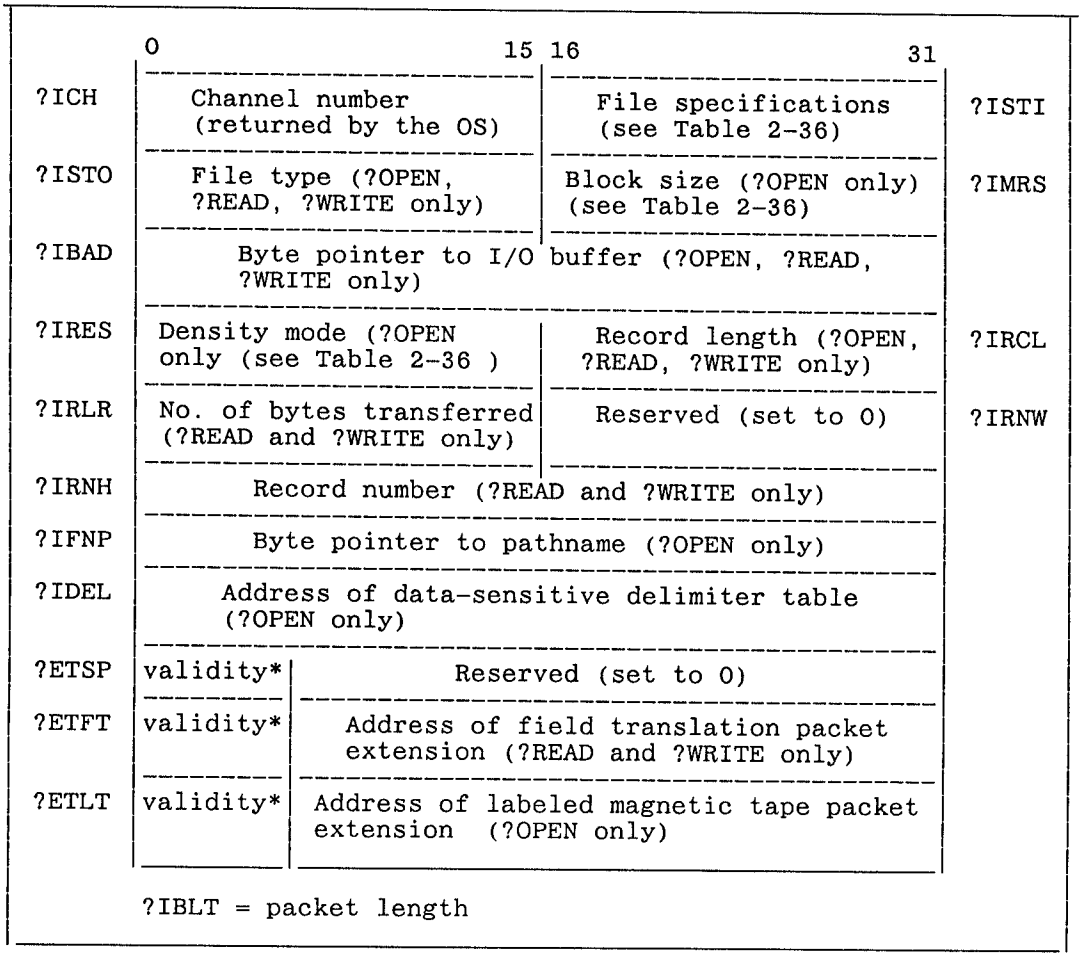


Figure 2-58. Standard I/O Packet for ?READ, ?WRITE, ?OPEN and ?CLOSE

By default, the operating system points to the next record (that is, the operating system increments the file pointer) when you read or write a record. However, there are two ways to reposition the file pointer:

- Issue the ?SPOS system call.
- Use offset ?IRNH in combination with the file pointer position parameter, ?IPST, in offset ?ISTI of the ?READ/?WRITE packet. (See Table 2-36 for a summary of this method.)

Table 2-36. Positioning the File Pointer

Parameter ?IPST	Offset ?IRNH	File Pointer Position
set	0	Beginning of file.
set	-1	End of file.
0	0	Next sequential record in file (relative to its current position).

Notes

- See the descriptions of ?SPOS and ?READ in this chapter for more information on setting the file-pointer position.

?GPRNM

Gets a program's pathname.

?GPRNM
error return
normal return

Input

AC0	One of the following: • PID of the target process • −1 to get the pathname of the calling process's program
AC1	Reserved (Set to 0.)
AC2	Byte pointer to a 256-byte buffer

Output

AC0	Unchanged
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERPRH	Attempt to access process not in hierarchy
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?GPRNM to get the pathname of the calling process's program or the pathname of another process.

What It Does

?GPRNM returns the complete pathname (starting at the root) of the disk file that was originally loaded into the target process's logical address space.

Before you issue ?GPRNM, perform the following steps:

1. Load AC0 with the PID of the target process, or with −1 to get the program name of the calling process.
2. Load AC2 with a byte pointer to a receive buffer you have set aside for the pathname.

Licensed Material - Property of Data General Corporation

?GRAPHICS

Manipulates pixel maps.

?GRAPHICS *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?GRAPHICS packet

Error Codes Returned in AC0

ERIFD	Illegal function for device
ERIGP	Invalid graphics parameter
ERIWR	Invalid window reference
ERPKT	Invalid packet ID
ERRVN	A reserved value was not set to 0
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid address passed as a system call argument
ERWNE	The window you specified does not exist

Error codes from the file system

Why Use It?

?GRAPHICS lets you manipulate pixel maps. You can use ?GRAPHICS to create, open, and close a pixel map. You can also use ?GRAPHICS to manipulate the palette associated with a pixel map, to copy a pixel map in and out of your program's address space, and to set a clip rectangle in a pixel map.

?GRAPHICS is useful only if your program will run on a graphics terminal in an AOS/VS or AOS/DVS windowing environment. The call applies to one graphics window or pixel map at a time. For example, to find out the status of every pixel map belonging to your program, you must issue the ?GRAPHICS call repeatedly, specifying a different pixel map each time.

?GRAPHICS is a multifunctional system call. To perform a particular function, you set the offset ?GRAPH_PKT.FUNC (in the main ?GRAPHICS packet) to the function code you want. Table 2-37 lists the valid function codes and the functions they perform.

?GRAPHICS Continued

Table 2-37. ?GRAPHICS Function Codes

Function Code	What It Lets You Do
?GRAPH_OPEN_WINDOW_PIXELMAP	Open a graphics window's pixel map for graphics output. (See the section "Opening a Graphics Window's Pixel Map.")
?GRAPH_CREATE_MEMORY_PIXELMAP	Create a pixel map in memory, and open it for graphics output. (See the section "Creating a Pixel Map in Memory.")
?GRAPH_CLOSE_PIXELMAP	Close a pixel map you previously opened using ?GRAPH_CREATE_MEMORY_PIXELMAP or ?GRAPH_OPEN_WINDOW_PIXELMAP. (See the section "Closing a Pixel Map.")
?GRAPH_PIXELMAP_STATUS	Get status of a pixel map. (See the section "Getting the Status of a Pixel Map.")
?GRAPH_SET_CLIP_RECTANGLE	Define a clip rectangle on a pixel map; enable or disable the clip rectangle. (See the section "Setting a Clip Rectangle.")
?GRAPH_MAP_PIXELMAP	Copy a pixel map into your program's address space so that you can then use standard MV instructions on it. (See the section "Transferring Data from Pixel Maps to Disk Files.")
?GRAPH_UNMAP_PIXELMAP	Remove a pixel map from your program's address space, so that you can then use graphics instructions on it. (See the section "Transferring Data from Pixel Maps to Disk Files.")
?GRAPH_WRITE_PALETTE	Define colors in the palette associated with a pixel map. (See the section "Writing to a Palette.")
?GRAPH_READ_PALETTE	Find out what colors are stored in the palette associated with a pixel map. (See the section "Reading a Palette.")

The Main ?GRAPHICS Packet

When you issue a ?GRAPHICS call, you set up a main packet (common to all functions of the call) and a subpacket (unique for each function). Most functions require subpackets. The structure of the main packet appears in Figure 2–59; we cover the subpackets later, when we discuss each function. Table 2–38 details the contents of each offset in the main packet.

The ?GRAPHICS call applies to one pixel map at a time. For most functions, you specify the target pixel map by supplying a pixel map ID in the main packet. (Functions ?GRAPH_OPEN_WINDOW_PIXELMAP and ?GRAPH_CREATE_MEMORY_PIXELMAP are exceptions which we explain later.)

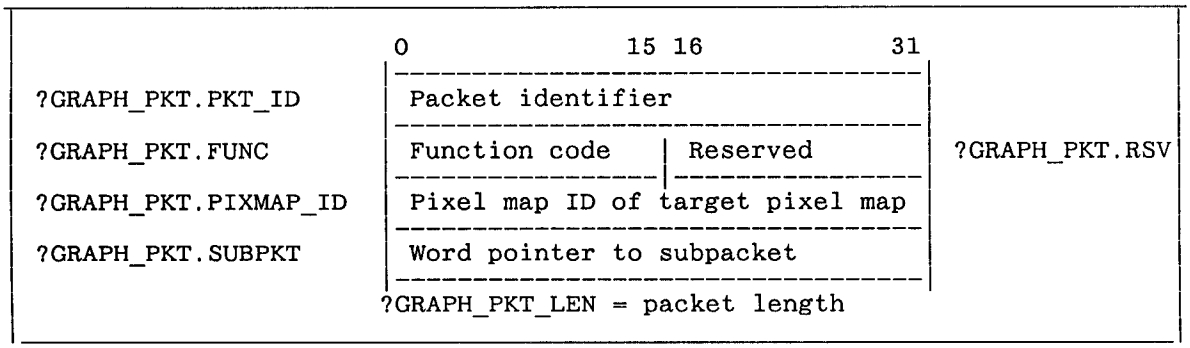


Figure 2–59. Structure of the ?GRAPHICS Main Packet

Table 2-38. Contents of the ?GRAPHICS Main Packet

Offset	Contents
?GRAPH_PKT.PKT_ID (double word)	Packet identifier; set to ?GRAPH_PKT_PKTID.
?GRAPH_PKT.FUNC	Code for the function you want. You must always supply a function code. (Function codes are listed in Table 2-37.)
?GRAPH_PKT.RSV	Reserved. (Set to 0.)
?GRAPH_PKT.PIXMAP_ID (double word)	Pixel map ID of the target pixel map. You must supply a pixel map ID for all functions except ?GRAPH_CREATE_MEMORY_PIXELMAP and ?GRAPH_OPEN_WINDOW_PIXELMAP. On these functions, the operating system returns a pixel map ID; set this offset to 0.
?GRAPH_PKT.SUBPKT (double word)	Word pointer to the subpacket for the function. If the function does not require a subpacket, set this offset to 0.

Opening a Graphics Window's Pixel Map

The virtual terminal associated with a graphics window is a pixel map, which can accept only output in the form of graphics instructions. When you issue graphics instructions, you specify the destination pixel map by a pixel map ID (which you can think of as a "graphics output channel"). A single graphics window can have more than one pixel map ID at a time, since you can open its pixel map as many times as you want.

NOTE: Before you open a graphics window, you must first create it using the ?WIN_CREATE_WINDOW function of ?WINDOW.

To get a pixel map ID for a graphics window, issue the ?GRAPHICS call with function code ?GRAPH_OPEN_WINDOW_PIXELMAP. In the main packet, set offset ?GRAPH_PKT.PIXMAP_ID to 0; the operating system returns the pixel map ID in this field.

In the subpacket, specify the graphics window you want to open. You can specify the window in one of three ways:

- Channel number — When you open a graphics window for input (using ?OPEN), the operating system returns a channel number. (Note that you can use the channel number for input only; you need a pixel map ID to send output to a graphics window.)
- Window pathname — When you create a window, you give it a window name. A window's pathname takes the form @PMAPn:windowname, where @PMAPn is the name of the pixel-mapped console that contains the window. The window pathname must end in a null character, <0>; it can be a maximum of ?MXPL characters long (including the null terminator).
- Window ID number — When you create a window, the operating system returns the window ID number in the main ?WINDOW packet.

Fill in the appropriate offset in the subpacket (channel number, pathname or ID number), and set the other two window specification offsets to zero. To indicate which method you are using to specify the window, you must set the appropriate flag in flag word ?GRAPH_OPEN.FLAGS. You can use only one method at a time.

Figure 2-60 shows the subpacket structure; Table 2-39 details the contents of the subpacket.

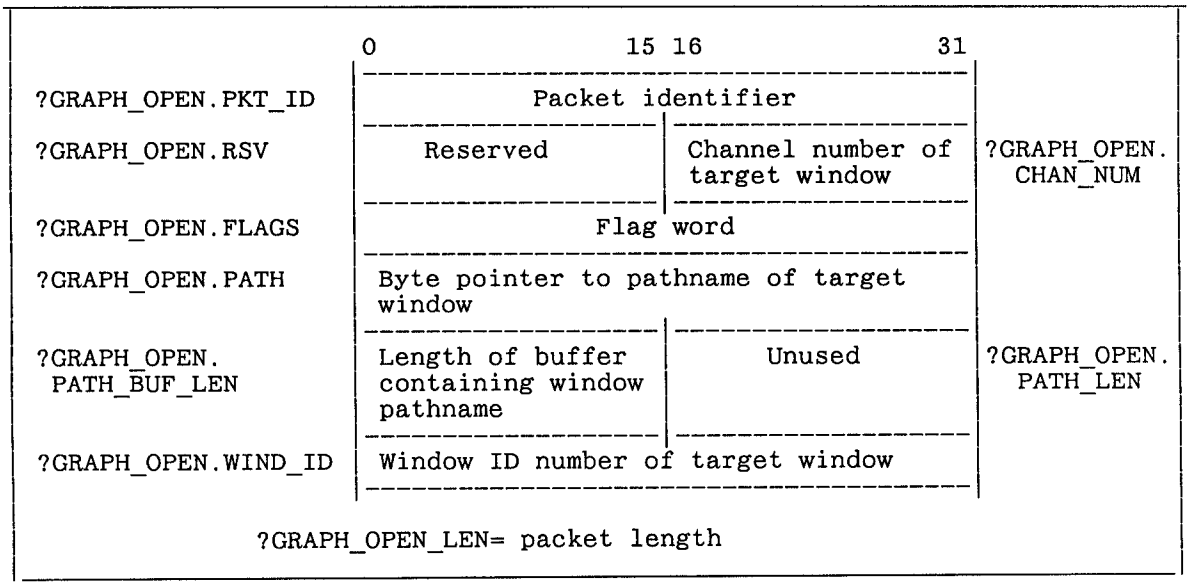


Figure 2-60. Structure of the ?GRAPH_OPEN_WINDOW_PIXELMAP Subpacket

Table 2-39. Contents of the ?GRAPH_OPEN_WINDOW_PIXELMAP Subpacket

Offset	Contents
?GRAPH_OPEN.PKT_ID (double word)	Packet identifier; set to ?GRAPH_OPEN_PKTID.
?GRAPH_OPEN.RSV	Reserved. (Set to 0.)
?GRAPH_OPEN.CHAN_NUM	Channel number of target graphics window. When specifying the window by its channel number, you must also set flag ?GRAPH_OPEN.FLAGS.IN_CHAN in flag word ?GRAPH_OPEN.FLAGS. Set this offset to 0 when specifying the target window by its pathname or window ID.
?GRAPH_OPEN.FLAGS (double word)	Flag word ?GRAPH_OPEN.FLAGS.IN_PATH -- Set this flag when specifying the target window by its pathname. ?GRAPH_OPEN.FLAGS.IN_WIND_ID -- Set this flag when specifying the target window by its window ID. ?GRAPH_OPEN.FLAGS.IN_CHAN -- Set this flag when specifying the target window by its channel number.
?GRAPH_OPEN.PATH (double word)	Byte pointer to pathname of target window. When specifying the window by its pathname, you must also set flag ?GRAPH_OPEN.FLAGS.IN_PATH in flag word ?GRAPH_OPEN.FLAGS. Set this offset to 0 when specifying the target window by its channel number or window ID.
?GRAPH_OPEN.PATH_BUF_LEN	Length of the buffer containing the window pathname. The buffer length must include the null terminator; the maximum buffer length is ?MXPL.
?GRAPH_OPEN.PATH_LEN	Unused. (Set to 0.)
?GRAPH_OPEN.WIND_ID (double word)	Window ID number of target window. When specifying the window by its window ID, you must also set flag ?GRAPH_OPEN.FLAGS.IN_WIND_ID in flag word ?GRAPH_OPEN.FLAGS. Set this offset to 0 when specifying the target window by its pathname or channel number.

Creating a Pixel Map in Memory

You use the ?GRAPH_CREATE_MEMORY_PIXELMAP function to create a pixel map in memory, and to open it for graphics output. Although such a pixel map is not visible on the physical screen, you can still perform graphics instructions on it, copy its contents to a graphics window's pixel map, and perform ?GRAPHICS functions on it.

In the main packet, set offset ?GRAPH_PKT.PIXMAP_ID to 0; the operating system returns a pixel map ID when it opens the newly created pixel map.

Your entries in the subpacket determine the attributes (width, height and pixel depth) of the new pixel map. Figure 2-61 shows the subpacket structure; Table 2-40 details its contents.

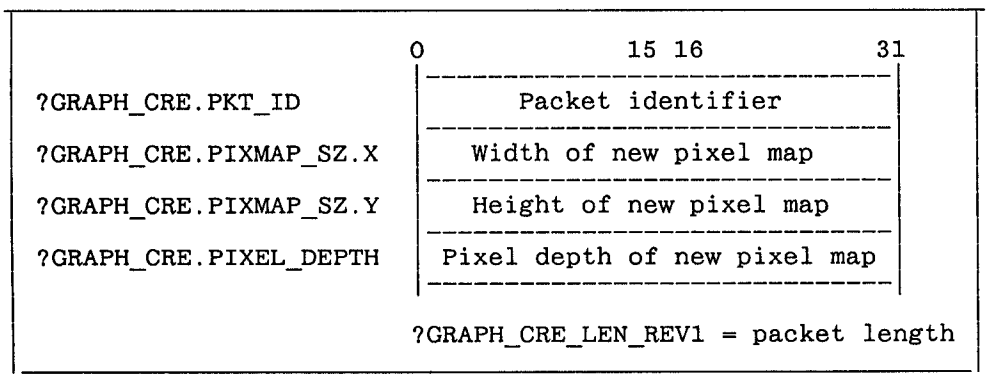


Figure 2-61. Structure of the ?GRAPHICS_CREATE_MEMORY_PIXELMAP Subpacket

Table 2-40. Contents of the ?GRAPH_CREATE_MEMORY_PIXELMAP Subpacket

Offset	Content
?GRAPH_CRE.PTK_ID (double word)	Packet identifier. Set to ?GRAPH_CRE_PKTID_REV1.
?GRAPH_CRE.PIXMAP_SZ.X (double word)	Width of pixel map (in pixels).
?GRAPH_CRE.PIXMAP_SZ.Y (double word)	Height of pixel map (in pixels).
?GRAPH_CRE.PIXEL_DEPTH (double word)	Pixel depth of pixel map (in bits).

Licensed Material - Property of Data General Corporation

Closing a Pixel Map

You use the ?GRAPHICS function ?GRAPH_CLOSE_PIXELMAP to close an open pixel map. The open pixel map can be a memory pixel map, or the pixel map associated with a graphics window.

All memory pixel maps are open: when you create a memory pixel map, the operating system automatically opens it. When you close a memory pixel map, the operating system automatically deletes it.

A graphics window's pixel map is open only if you have issued the ?GRAPH_OPEN_WINDOW_PIXELMAP function for that window. You can open a graphics window's pixel map more than once; each time, the operating system returns a separate pixel map ID. When you close a graphics window's pixel map, the operating system closes the pixel map ID you specified. If you open a graphics window's pixel map more than once, you must issue a separate ?GRAPH_CLOSE_PIXELMAP call for each pixel map ID. (To delete a graphics window's pixel map, you must delete the window itself; use the ?WINDOW call, function code ?WIN_DELETE_WINDOW.)

In the main packet, specify the pixel map ID of the pixel map you want to close. Since there is no subpacket for this function, set ?GRAPH_PKT.SUBPKT to 0.

Getting the Status of a Pixel Map

You use ?GRAPH_PIXELMAP_STATUS to get current information about a pixel map. You can get information about either a memory pixel map or a graphics window's pixel map.

In the main packet, specify the pixel map ID of the pixel map you want. In the subpacket, supply a packet identifier; set all other offsets to 0. The operating system returns the following information in the subpacket:

- Width and height of the pixel map.
- Pixel depth.
- The coordinates of the clip rectangle's origin.
- Dimensions of the clip rectangle.
- Status of the clip rectangle: enabled or disabled.

Figure 2–62 shows the subpacket structure; Table 2–41 details its contents.

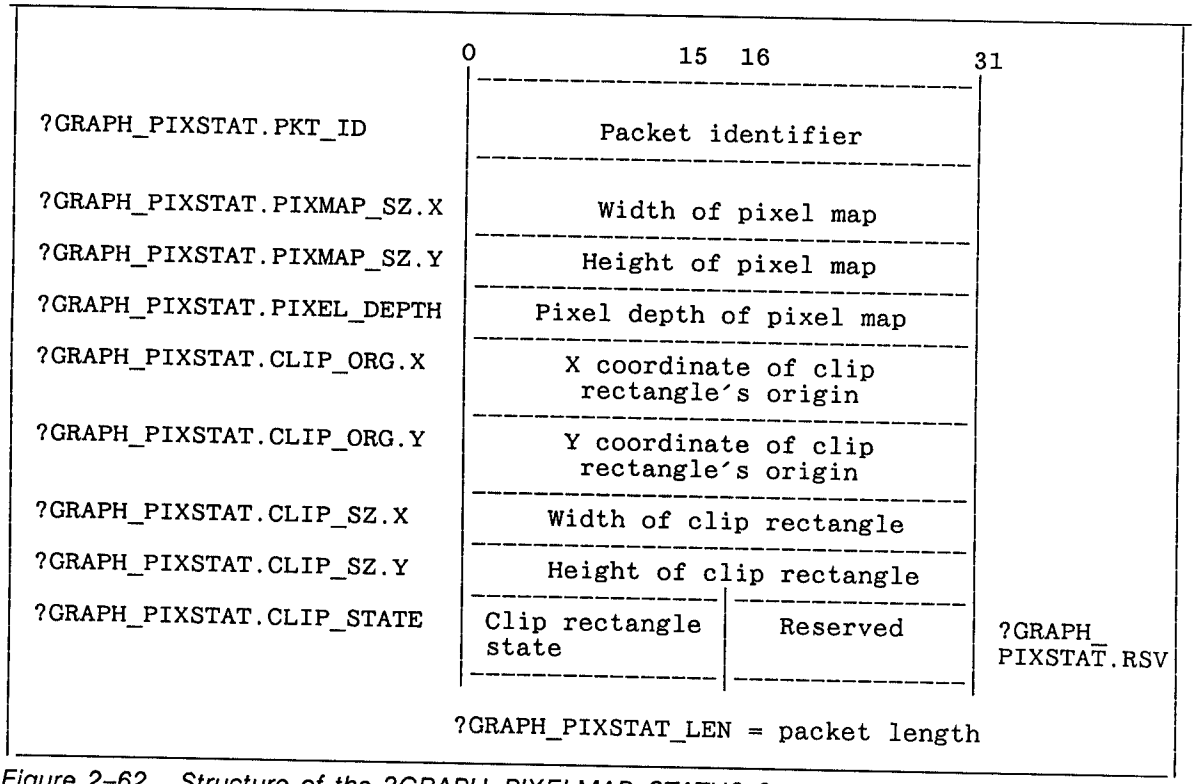


Figure 2–62. Structure of the ?GRAPH_PIXELMAP_STATUS Subpacket

Licensed Material - Property of Data General Corporation

Table 2-41. Contents of the ?GRAPH_PIXELMAP_STATUS Subpacket

Offset	Contents
?GRAPH_PIXSTAT.PKT_ID (double word)	Packet identifier. Set to ?GRAPH_PIXSTAT_PKTID.
?GRAPH_PIXSTAT.PIXMAP_SZ.X (double word)	Width (in pixels) of pixel map. Returned by the operating system.
?GRAPH_PIXSTAT.PIXMAP_SZ.Y (double word)	Height (in pixels) of pixel map. Returned by the operating system.
?GRAPH_PIXSTAT.PIXEL_DEPTH (double word)	Pixel depth of pixel map (number of bits per pixel).
?GRAPH_PIXSTAT.CLIP_ORG.X (double word)	X coordinate of clip rectangle's origin (in pixels relative to origin of pixel map).
?GRAPH_PIXSTAT.CLIP_ORG.Y (double word)	Y coordinate of clip rectangle's origin (in pixels relative to origin of pixel map)
?GRAPH_PIXSTAT.CLIP_SZ.X (double word)	Width (in pixels) of clip rectangle.
?GRAPH_PIXSTAT.CLIP_SZ.Y (double word)	Height (in pixels) of clip rectangle.
?GRAPH_PIXSTAT.CLIP_STATE	The current state of the clip rectangle. ?GRAPH_RECT_STATE_ENABLE -- Indicates that the clip rectangle is currently enabled. ?GRAPH_RECT_STATE_DISABLE -- Indicates that the clip rectangle is currently disabled.
?GRAPH_PIXSTAT.RSV	Reserved. (Set to 0.)

Setting the Clip Rectangle

You use the ?GRAPH_SET_CLIP_RECTANGLE function to specify a rectangular portion of a pixel map as a clip rectangle. When a clip rectangle is in effect, you can still specify any location on the pixel map as an argument to a graphics instruction. However, any changes affect only the portion of the pixel map within the rectangle.

Every pixel map has a clip rectangle. Initially, the clip rectangle has the same size and origin as the pixel map, and clipping is disabled. The ?GRAPH_SET_CLIP_RECTANGLE function lets you set the clip rectangle's origin, width and height, and lets you disable or re-enable clipping.

In the main packet, specify the pixel map ID of the pixel map for which you want to set the clip rectangle.

Your entries in the subpacket determine which attributes of the clip rectangle (its size, origin, or state) will change. To reset a particular attribute, place the new value in the appropriate offset; in the flag word ?GRAPH_SET_CLIP.FLAGS, set the flag that corresponds to that offset. If you don't want to change a particular attribute, set both the offset and the corresponding flag to 0. Figure 2-63 shows the subpacket structure; Table 2-42 details its contents.

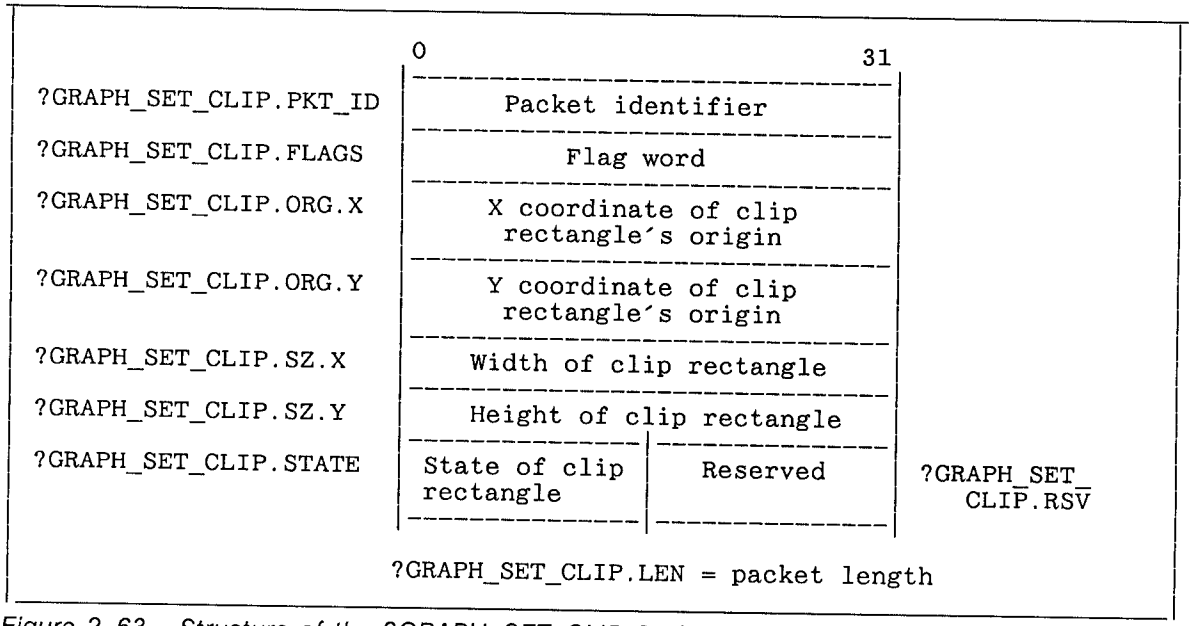


Figure 2-63. Structure of the ?GRAPH_SET_CLIP_RECTANGLE Subpacket

Table 2-42. Contents of the ?GRAPH_SET_CLIP.RECTANGLE Subpacket

Offset	Contents
?GRAPH_SET_CLIP.PKT_ID (double word)	Packet identifier. Set this offset to ?GRAPH_SET_CLIP_PKTID.
?GRAPH_SET_CLIP.FLAGS (double word)	Flag word. For each clip rectangle attribute you want to change, you must set the corrresponding flag. ?GRAPH_SET_CLIP.FLAGS.ORG_X -- You must set this flag if you are changing the X coordinate of the clip rectangle's origin. ?GRAPH_SET_CLIP.FLAGS.ORG_Y -- You must set this flag if you are changing the Y coordinate of the clip rectangle's origin. ?GRAPH_SET_CLIP.FLAGS.SZ_X -- You must set this flag if you are changing the width of the clip rectangle. ?GRAPH_SET_CLIP.FLAGS.SZ_Y -- You must set this flag if you are changing the height of the clip rectangle. ?GRAPH_SET_CLIP.FLAGS.STATE -- You must set this flag if you are changing the state of the clip rectangle.

(Continues)

Table 2-42. Contents of the ?GRAPH_SET_CLIP.RECTANGLE Subpacket

Offset	Contents
?GRAPH_SET_CLIP.ORG.X (double word)	X coordinate of clip rectangle's origin (in pixels, relative to origin of pixel map). If you supply a value here, you must also set the flag ?GRAPH_SET_CLIP.FLAGS.ORG_X. If you are not changing the X coordinate, set this offset to 0, and do not set the corresponding flag.
?GRAPH_SET_CLIP.ORG.Y (double word)	Y coordinate of clip rectangle's origin (in pixels, relative to origin of pixel map). If you supply a value here, you must also set the flag ?GRAPH_SET_CLIP.FLAGS.ORG_Y. If you are not changing the Y coordinate, set this offset to 0, and do not set the corresponding flag.
?GRAPH_SET_CLIP.SZ.X (double word)	Width of clip rectangle (in pixels). If you supply a value here, you must also set the flag ?GRAPH_SET_CLIP.FLAGS.SZ.X. If you are not changing the width, set this offset to 0, and do not set the corresponding flag.
?GRAPH_SET_CLIP.SZ.Y (double word)	Height of clip rectangle (in pixels). If you supply a value here, you must also set the flag ?GRAPH_SET_CLIP.FLAGS.SZ.Y. If you are not changing the width, set this offset to 0, and do not set the corresponding flag.
?GRAPH_SET_CLIP.STATE	State of clip rectangle. To change the state, place one of the following values here, and set the flag ?GRAPH_SET_CLIP.FLAGS.STATE. If you are not changing the state, set this offset to 0, and do not set the corresponding flag. ?GRAPH_RECT_STATE_ENABLE -- Enables the clip rectangle. ?GRAPH_RECT_STATE_DISABLE -- Disables the clip rectangle.
?GRAPH_SET_CLIP.RSV	Reserved. (Set to 0.)

(Concluded)

Transferring Data Between Pixel Maps and Disk Files

To transfer information between a disk file and a pixel map,

1. Issue the ?GRAPHICS call with function code ?GRAPH_MAP_PIXELMAP to map an existing memory pixel map into your program's address space. (We explain this step in more detail in the section "To Map a Pixel Map Into Your Program's Address Space" below.)

2. Use I/O system calls, such as ?READ and ?WRITE, to transfer data between the disk file and the address space.

3. Issue ?GRAPHICS with function code ?GRAPH_UNMAP_PIXELMAP to unmap the pixel map from the address space. This returns the contents of the address space to pixel map format; you can then use graphics instructions on that pixel map. (We explain this step in more detail in the section "To Unmap a Pixel Map from Your Program's Address Space" below.)

To Map a Pixel Map into Your Program's Address Space

You use the ?GRAPH_MAP_PIXELMAP function to load a pixel map into your program's address space. In the main packet, specify the pixel map ID of the pixel map you want.

The operating system calculates where to place the pixel map in your address space, and returns the address of the pixel map in the subpacket (offset ?GRAPH_MAP_PMAP.ADDR). It also returns information that lets you convert pixel map coordinates into word and bit addresses so that you can perform data transfer instructions accurately.

Figure 2-64 shows the subpacket structure; Table 2-43 details its contents.

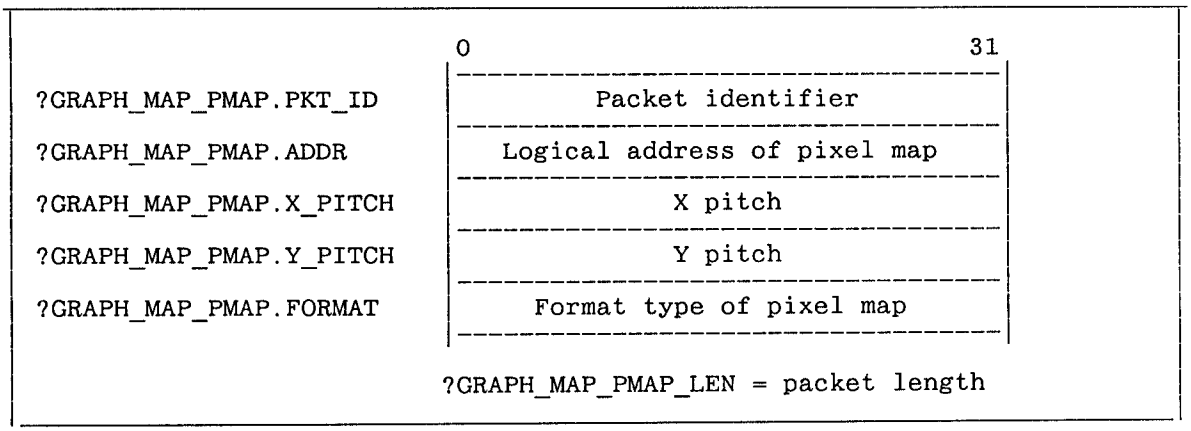


Figure 2-64. Structure of the ?GRAPH_MAP_PIXELMAP Subpacket

Table 2-43. Contents of the ?GRAPH_MAP_PIXELMAP Subpacket

Offset	Contents
?GRAPH_MAP_PMAP.PKT_ID (double word)	Packet identifier. Set to ?GRAPH_MAP_PMAP_PKTID.
?GRAPH_MAP_PMAP.ADDR (double word)	Logical address of pixel map in your program's address space (returned by the operating system).
?GRAPH_MAP_PMAP.X_PITCH	The number of bits occupied by a pixel; not necessarily the same as pixel depth. (Returned by the operating system.)
?GRAPH_MAP_PMAP.Y_PITCH	The number of pixels on a line; not necessarily the same as the width of the pixel map. (Returned by the operating system.)
?GRAPH_MAP_PMAP.FORMAT (double word)	Pixel map format type (returned by the operating system).
	?GRAPH_MAP_PMAP.FORMAT.PACKED_PIXEL -- Standard format for Data General pixel maps.

To get the bit offset (from the beginning of the mapped pixel map) of a particular pixel, use the following formula:

Bit offset = (y_coordinate * X_PITCH * Y_PITCH) + (X_coordinate * X_PITCH)

Licensed Material - Property of Data General Corporation

To Unmap a Pixel Map from Your Program's Address Space

Once you have mapped a pixel map into your program's address space, you cannot use graphics instructions on that pixel map until you have unmapped it.

To unmap a pixel map, issue ?GRAPHICS with function code ?GRAPH_UNMAP_PIXELMAP. The operating system then unmaps that pixel map from your program's address space to the pixel map's original location. Any data you wrote to the pixel map while it was in your address space will appear in the unmapped pixel map, and you can then use graphics instructions on that pixel map.

In the main packet, specify the pixel map ID of the target pixel map. There is no subpacket for the ?GRAPH_UNMAP_PIXELMAP function; set offset ?GRAPH_PKT.SUBPKT to 0.

Writing to a Palette

You use function ?GRAPH_WRITE_PALETTE to change the colors in the palette associated with a pixel map. You can change the color settings for one or more contiguous pixel values. For the set of pixel values you specify, you can change either the phase 0 setting, the phase 1 setting, or both. Follow these steps to change the colors in a palette:

1. Set up an array in memory to contain the color settings. If you are changing both the phase 0 and phase 1 settings, you must set up two separate arrays, one for each phase. The arrays must be the same size.

   There must be an entry in each array for each pixel value you want to change. An entry consists of four 32-bit unsigned fractions, one for each color level:

   <Red level> <Green level> <Blue level> <Grey level>

2. Issue the ?GRAPHICS call with function code ?GRAPH_WRITE_PALETTE. In the main packet, specify the pixel map ID of the pixel map whose palette you are changing. In the subpacket, you specify

   - The number of palette entries you want to write.
   - The palette index at which you want to start writing.
   - The addresses of the arrays you have stored in memory.

Figure 2–65 shows the subpacket structure.

The operating system then copies the array(s) you specify into the palette, starting at the specified palette index.

	0 31
?GRAPH_WRPAL.PKT_ID	Packet identifier; set to ?GRAPH_WRPAL_PKTID
?GRAPH_WRPAL.ARRAY_PKT_ID	Array packet identifier; set to ?GRAPH_WRPAL_ARRAY_PKTID
?GRAPH_WRPAL.FIRST_PAL_INDEX	Index of first palette entry to write (palette indices are zero-relative)
?GRAPH_WRPAL.ENTRY_CNT	Number of palette entries to write (entries must be consecutive)
?GRAPH_WRPAL.PH0_ARRAY	Word pointer to Phase 0 array (set to 0 if you are not changing the Phase 0 settings)
?GRAPH_WRPAL.PH1_ARRAY	Word pointer to Phase 1 array (set to 0 if you are not changing the Phase 1 settings)
	?GRAPH_WRPAL_LEN = packet length

Figure 2–65. Structure of the ?GRAPH_WRITE_PALETTE Subpacket

Reading from a Palette

You can use function ?GRAPH_READ_PALETTE to get the current color settings from the palette associated with a pixel map. This call copies a series of color settings for contiguous pixel values from the palette into memory. Follow these steps to determine the colors in a palette:

1. Allocate space in memory for each array you are copying. If you are copying both the Phase 0 and Phase 1 settings, you must allocate space for two separate arrays (one for each phase). Each array needs as much space as the number of entries you will copy, multiplied by the size of each entry (four double words).

2. Issue the ?GRAPHICS call with function code ?GRAPH_READ_PALETTE. In the main packet, specify the pixel map ID of the pixel map whose palette you are reading. In the subpacket, you specify

   - The number of palette entries you are reading.
   - The palette index at which you want to start reading.
   - The destination memory address for each array.

The operating system then copies the palette entries you specified into the space you allocated in memory. The copied entries are set up as arrays, one for each phase. Each array contains an entry for each pixel value you requested. Each entry consists of four double words, each of which denotes a color level:

<Red level> <Green level> <Blue level> <Grey level>

Figure 2-66 shows the subpacket structure.

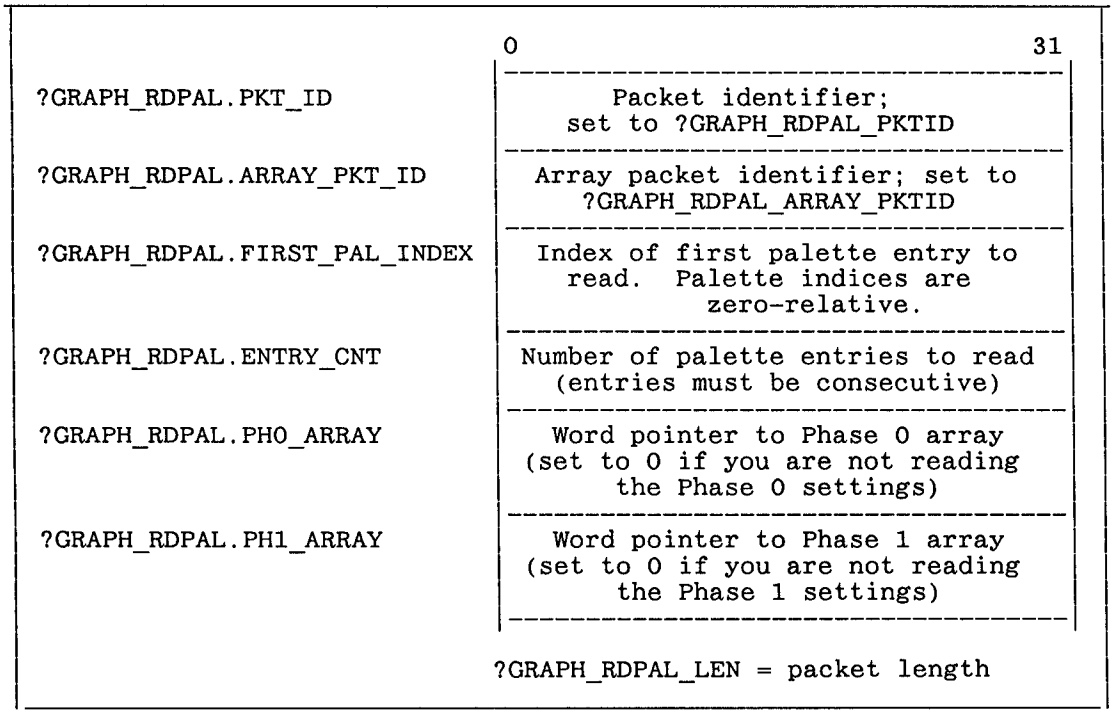


Figure 2-66. Structure of the ?GRAPH_READ_PALETTE Subpacket

?GRNAME Returns complete pathname of generic file.

?GRNAME
error return
normal return

Input

AC0	Byte pointer to the input generic filename
AC1	Byte pointer to the buffer
AC2	Length of the buffer in bytes

Output

AC0	Unchanged
AC1	Unchanged
AC2	Length of the name in byte, excluding the null terminator

Error Codes in AC0

ERFDE	File does not exist (The file may be a generic file, but no generic-to-real connection exists.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?GRNAME allows you to determine what file is associated with a particular generic file. Also, you can use ?GRNAME to return the complete pathname of a nongeneric pathname fragment in the same way that you would use ?GNAME.

What It Does

?GRNAME returns the complete pathname associated with a generic file.

If the input file is a generic file and there is no connected "real" file, then the operating system returns error code ERFDE in AC0 and sets Bit 0 of AC2 to 0. If the file is a generic file that is connected to a "real" file, then the operating system returns the full pathname of the connected file as if a ?GNAME were performed on that pathname. In this case, the operating system sets Bit 0 of AC2 to 1.

Notes

- You cannot use ?GNAME to obtain the "true" pathname of a generic file. For example, ?GNAME would return a complete pathname of :PER:DATA for the input pathname @DATA. However, the complete pathname of the file is actually :UDD:USER:DATA. In this case, ?GRNAME would return :UDD:USER:DATA.
- If you place in AC0 a byte pointer to the text string @CONSOLE, then ?GRNAME places :PER:CONSOLE in the buffer whose byte pointer is in AC1.
- See the description of ?GNAME in this chapter.

?GSHPT Lists the current shared partition size.

?GSHPT
error return
normal return

Input

None

Output

AC0 Page number of the first shared page in the calling process's current ring

AC1 Total number of shared pages in the calling process's current ring

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?GSHPT lets you determine the amount of logical shared memory the operating system has allocated to your process. Also, you can use ?GSHPT with ?SSHPT, which establishes a new shared partition in the caller's logical address space.

What It Does

?GSHPT returns the total number of shared pages in the caller's current ring, and the logical page number of the first shared page. ?GSHPT has no input requirements.

The operating system calculates the logical page numbers and returns them in AC0 and AC1. These page numbers range from 1 through 262,143 shared pages for Ring 7 and from 1 through 512 shared pages for Rings 4 through 6. (Page zero is always unshared.) If the caller's logical address space contains no shared pages, the operating system returns 0 to both AC0 and AC1.

Notes

- See the description of ?SSHPT in this chapter.

?GSID

Gets the system identifier.

?GSID
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Byte pointer to a 32-byte receive buffer for the system identifier

Output

AC0	Undefined
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERMPR	System call parameter address error
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?GSID to get the system identifier for a particular system. This can be useful if you don't know what system you are on.

What It Does

?GSID returns the current system identifier to a receive buffer in your logical address space. The system identifier is an ASCII character string 32 bytes long. The string returned by ?GSID is exactly the same as the one set by ?SSID. By convention, anything appearing after the first null (if any) in the system identifier is ignored by most processes. (If the identifier is less than 32 characters long, the system puts a null terminator on the end; if the identifier is 32 characters long, no null is added.) The operator process (PID 2) can define the system identifier with the ?SSID system call.

Before you issue ?GSID, set up a 32-byte receive buffer for the identifier in your address space, and load AC2 with a byte pointer to the buffer.

Notes

- See the description of ?SSID in this chapter.

Licensed Material - Property of Data General Corporation

?GTACP — Gets access control privileges.

?GTACP
error return
normal return

Input

AC0	Byte pointer to the target file's pathname
AC1	One of the following: • Byte pointer to the target username • -1 to get the caller's access privileges
AC2	Reserved (Set to 0.)

Output

AC0	One or more of the following access privilege masks: ?FAOB Owner access ?FAWB Write access ?FAAB Append access ?FARB Read acess ?FAEB Execute access
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERPRV	Caller not privileged for this action
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

Because ?GTACP returns the ACL for a specific file and username, you can issue ?GTACP before you issue ?SACL, which sets the current ACL for a file, or before you issue ?DACL, which sets the default ACL. If you are in Superuser mode, ?GTACP allows you to find out if a given user has access to a particular file.

What It Does

?GTACP returns bit masks in AC0 that indicate the target username's access privileges to the target file. To issue ?GTACP against another username, the calling process must be in Superuser mode; otherwise, ?GTACP fails on error code ERPRV.

Before you issue ?GTACP, specify the target file's pathname and the target username in your logical address space. Load AC0 with a byte pointer to the pathname, and load AC1 with a byte pointer to the username. The user parameter files, PARU.32.SR and PARU.16.SR, define the privilege bits as follows:

Mask	Bit Value	Meaning
?FACO	1B(?FAOB)	Owner access
?FACW	1B(?FAWB)	Write access
?FACA	1B(?FAAB)	Append access
?FACR	1B(?FARB)	Read access
?FACE	1B(?FAEB)	Execute access

Notes

- See PARU.32.SR or PARU.16.SR.

?GTMES

Gets a CLI message.

?GTMES *[packet address]*
error return
normal return

Input

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

Output

AC0	Dependent on packet's input values (See Table 2–44.)
AC1	Dependent on packet's input values (See Table 2–44.)
AC2	Address of the ?GTMES packet

Error Codes in AC0

ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?GTMES allows you to access the initial IPC message that the CLI sends when it creates a new process. This message is an edited version of the CLI command used to create the process. Depending on your input specifications, you use ?GTMES to get a specific argument in the CLI command line and to determine which switches, if any, modify it. ?GTMES also returns the message that another father process sends when it creates a son with the ?PROC system call.

What It Does

When you create a process by executing a CLI command, the CLI sends the new process an initial message that contains an edited version of the CLI command. ?GTMES gives a CLI son process access to that message or, depending on your input specifications, access to

- The length of the message, in words.
- The number of arguments in the CLI command (argument count) and any simple or keyword switches used to modify a particular argument. A keyword switch is a two–part switch of the form

  /keyword = value.

  For example, /L=filename is a keyword switch.
- One of the CLI command line arguments, minus the switches.

?GTMES Continued

Figure 2-67 shows the structure of the ?GTMES packet, and Table 2-44 describes each offset and parameter in the packet.

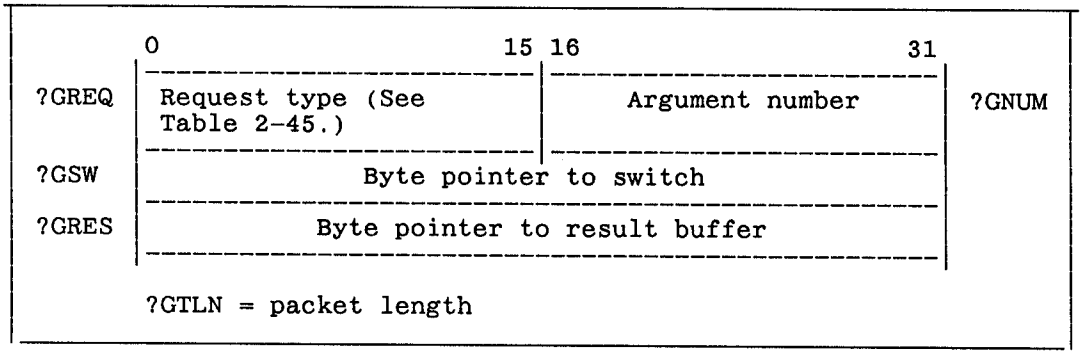


Figure 2-67. Structure of ?GTMES Packet

Most of the ?GTMES parameters apply to CLI messages. However, offset ?PIPC in the ?PROC packet passes an IPC header from the ?PROC caller to the son, which allows you to use ?GTMES to obtain an IPC message sent by father process (other than the CLI) when it creates a son.

Offset ?GREQ describes the ?GTMES request; that is, the kind of information you want the operating system to return. Choose only one request type for offset ?GREQ. Table 2-45 lists the possible request types and their meanings.

Table 2-44. Contents of ?GTMES Packet*

Offset	Contents
?GREQ	Request type: specifies the information you want the OS to return. (See Table 2-45 for request types.)
?GNUM	Argument in the CLI command line; specify the argument you want by its relative position in the command line. For example, ?GNUM = 0 specifies the first argument; that is, program name.
?GSW (double word)	Byte pointer to a simple or keyword switch you want to test for. (Use ?GSW only if request type is ?GTSW; see Table 2-45.) A keyword switch can be no longer than 32 bytes, including the terminating null. DEFAULT = 0 (do not test for a switch).
?GRES (double word)	Byte pointer to buffer you want to receive the results. (The OS terminates the results string with a null.) DEFAULT = -1 (pass results to AC0 and AC1 only) (See Table 2-46.)

* There is no default unless otherwise specified.

Table 2-45. Input Parameters for Offset ?GREQ (Request Types)

Request	Meaning
?GMES	Copy the entire message to the message buffer specified in offset ?GRES (unless this offset contains -1). Return the length of the message (number of words) to AC0, and the message flag, if present, to AC1. The message flag is ?GFCF Message is in CLI format (the first argument is program name).
?GCMD	Read the edited CLI command line into the buffer specified in offset ?GRES (unless these offsets contain -1). Return the byte length of the edited command line to AC1.
?GCNT	Get the argument count (number of arguments in the CLI command line, excluding the program name) and return this value to AC0.
?GARG	Copy the command line argument specified in ?GNUM, minus switches, to the ?GRES buffer area (if that area exists). If the argument consists entirely of decimal digits, the OS converts it to binary, returns the results to AC1, and returns the argument's byte length in AC0. (The largest possible value is 65,535. If the argument value is larger than 65,535, the OS not only returns an error, but it also returns the number of characters in the argument to AC1.)
?GTSW	Test the simple or keyword switch referred to by ?GSW to see whether it modifies the argument cited in ?GNUM. Return the following test results to AC0: -1 switch not found. 0 switch found. >0 byte length of the keyword switch. If the OS finds a keyword switch, it returns that switch's value, terminated by a null, to the ?GRES buffer.

(Continues)

Table 2-45. Input Parameters for Offset ?GREQ (Request Types)

Request	Meaning
?GTSW (Cont.)	If the switch value consists entirely of decimal digits, the OS converts it to binary and returns the result to AC1. (The largest possible value is 65,535. If the keyword switch is larger than 65,535, the OS not only returns an error, but it also returns the number of characters in the keyword switch to AC1.) If the switch appears in the argument more than once, the OS returns information for the first occurrence only.
?GSWS	Examine the ?GNUM argument for simple switches or alphabetic keyword switches. Set corresponding low-order bits in AC0 and AC1 for each switch. For example: In AC0 1S16 means /A 1S17 means /B ... 1S31 means /P In AC1 1S16 means /Q 1S17 means /R ... 1S25 means /Z (Bits 26 through 31 in AC1 are always 0.)

(Concluded)

As Table 2-45 shows, several of the ?GTMES requests use the buffer area specified by offset ?GRES. If you choose the default buffer specification (by setting ?GRES to -1), the OS returns information only to AC0 and AC1. Table 2-46 summarizes the information returned to the buffer area and to AC0 and AC1.

?GTMES Continued

Table 2-46. Output from ?GTMES Requests

Request Type	AC0	AC1	?GRES Buffer
?GMES	Message Flag (if message is in CLI format).	Total word length of message.	Entire message.
?GCMD	Unchanged.	Byte length of CLI command line.	CLI command line
?GCNT	Number of arguments (excluding argument 0 and program name).	Unchanged.	N/A.
?GARG	Byte length of argument specified in ?GNUM (excluding terminating null).	Binary equivalent of argument, if argument is decimal value not greater than 65,535. If argument value is greater than 65,535, return number of characters in argument. If argument is non-numeric, -1.	Actual argument string.
?GTSW	Switch test results -1 No switch. 0 Simple switch. >0 Byte length of keyword switch.	Binary equivalent of keyword switch, if keyword switch is decimal value not greater than 65,535. If keyword switch is greater than 65,535, return number of characters in keyword switch. If keyword switch is nonnumeric, -1.	Actual keyword switch.
?GSWS	Switch value 1S16 = /A 1S17 = /B ... 1S31 = /P	Switch value 1S16 = /Q 1S17 = /R ... 1S25 = /Z (Bits 26-31 = 0)	N/A.

Licensed Material - Property of Data General Corporation

Request type ?GMES copies the entire message to the ?GRES area (if it exists) and copies the message length to AC1. If the message is in CLI format, this request type also passes the ?GFCF flag to AC0.

If you specify a buffer area and select a request type other than ?GMES, the CLI returns an edited version of the message with the following format:

- One comma separating each argument.
- The message word aligned and terminated by a null (or two nulls, if necessary to ensure an even number of bytes).
- The high-order bit of every byte set to 0 (not used for parity).
- Lowercase characters converted to uppercase characters.
- Without the following characters: space, <, >, [,], (,), ;, NEW LINE, carriage return, form feed, embedded nulls.

If you select request type ?GCMD and the CLI command was EXECUTE, XEQ DEBUG, PROCESS, or CHAIN, the operating system removes the command name from the edited message.

Request type ?GTSW tests to see if a simple or keyword switch modifies the argument in offset ?GNUM. Be sure to set ?GNUM to an argument number (that is, to the relative position of the argument) within the range 0 through n, where 0 is the first argument (program name, for CLI command), and n is the last argument. Use offset ?GSW as a byte pointer to the switch you wish to test. When you specify the switch, remove the initial backslash and terminate the switch with a null. For example, write /L as L<0>.

Request type ?GSWS also tests for the presence of a switch in the argument and, in addition, sets a corresponding bit in the low-order portion of either AC0 or AC1.

To find out whether or not the message is written in CLI format, select ?GMES as the request type and set ?GRES to -1 (no buffer area). Then examine the output value of AC0. If AC0 contains the message flag ?GCFC, the message is in CLI format.

Sample Packet

The following sample packet gets a filename argument:

```
PKT:    .BLK    ?GTLN           ;Allocate enough space for the packet
                                ;(packet length = ?GTLN).
        .LOC    PKT+?GREQ       ;Request type. (See Table 2-45.)
        .WORD   ?GARG           ;Get the command line argument
                                ;specified in ?GNUM.
        .LOC    PKT+?GNUM       ;Argument number.
        .WORD   -1              ;Specify argument number at ?GTMES
                                ;time.
        .LOC    PKT+?GSW        ;Byte pointer to switch you want to
                                ;test for.
        .DWORD  0               ;Not used. (You only use ?GSW when
                                ;the request type is ?GTSW.)
        .LOC    ?GRES           ;Byte pointer to buffer that receives
        .DWORD  -1              ;the results. Specify at ?GTMES time.
        .LOC    PKT+?GTLN       ;End of packet.
```

Notes

- See the description of ?PROC in this chapter.

?GTNAM

Returns symbol closest in value to specified input value.

?GTNAM
error return
normal return

Input

AC0	Input value
AC1	Contains the following: • Bits 16 through 23 contain the byte length of the receive buffer (specified in AC2) • Bits 24 through 31 contain the channel number of the .ST file
AC2	Byte pointer to a receive buffer for the symbol name

Output

AC0	Offset between the input value that you specify and the symbol in the receive buffer
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERICN	Invalid channel number
ERIRB	Insufficient room in buffer (The receive buffer is too small.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?GTNAM allows you to search the symbol table (.ST file) for a symbol that approximates a specific value, without writing a routine for this purpose. For example, you can use ?GTNAM in a utility program that references symbols in the .ST file. ?GTNAM lets you reference the .ST file without knowing its format.

Licensed Material - Property of Data General Corporation

?GTNAM Continued

What It Does

?GTNAM searches the symbol table (.ST file) and returns the symbol closest in value to the value you placed in AC0.

The OS returns the symbol name to the receive buffer that you specify in AC2, and also returns the offset between the input value and the symbol value to AC0.

Before you issue ?GTNAM, perform the following steps:

1. ?SOPEN your program's .ST file.
2. Set up a receive buffer for the symbol name and load its address into AC2.
3. Specify the input value in AC0.
4. Specify the length of the receive buffer and the channel number of the .ST file in AC1. (The ?SOPEN system call returns the channel number.)

The symbol that ?GTNAM returns can consist of up to 32 bytes. Thus, you should define a receive buffer of at least 32 bytes. If the operating system cannot find a symbol that is less than the input value, it returns a 32-byte null string to the receive buffer.

Notes

- See the descriptions of ?SOPEN and ?GTSVL in this chapter.

?GTOD

Gets the time of day.

?GTOD
error return
normal return

Input

None

Output

AC0 Seconds from 0 through 59

AC1 Minutes from 0 through 59

AC2 Hour from 0 through 23

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?GTOD to obtain the current time for output on file listings.

What It Does

?GTOD gets the current time from the system clock and returns it to the accumulators in standard notation. The value for the hour can range from 0 (midnight) through 23 (11 p.m.), and the values for seconds and minutes can range from 0 through 59.

?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

File System Error Codes

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.

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

Figure 2-68 shows the structure of the ?GTRUNCATE packet, and Table 2-47 describes its contents.

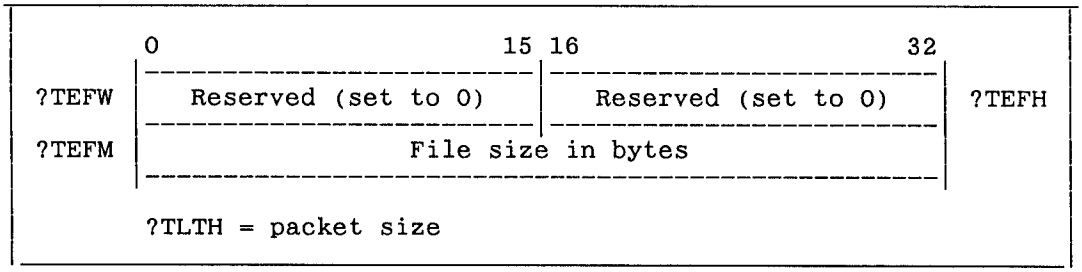


Figure 2-68. Structure of ?GTRUNCATE Packet

Table 2-47. Contents of ?GTRUNCATE Packet*

Offset	Contents
?TEFW	Reserved. (Set to 0.)
?TEFH	Reserved. (Set to 0.)
?TEFM (double word)	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.

?GTSVL

Gets the value of a user symbol.

?GTSVL
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Contains the following: • Bits 16 through 23 contain the byte length of the symbol name (specified in AC2) • Bits 24 through 31 contain the channel number of the .ST file
AC2	Byte pointer to the name of the target symbol

Output

AC0	Value of the target symbol
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERICN	Invalid channel number
ERSNF	Symbol not found in ?GTSVL system call (The symbol is not in the .ST file.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?GTSVL lets you retrieve the value of a particular user symbol, as recorded in your program's .ST file. Like ?GTNAM, ?GTSVL allows you to reference the .ST file without knowing its exact format.

Licensed Material - Property of Data General Corporation

What It Does

?GTSVL returns the symbol table value of a particular user-defined symbol. Before you issue ?GTSVL, perform the following steps:

1. ?SOPEN your program's .ST file (symbol table).
2. Specify the target symbol name in your logical address space and load AC2 with a byte pointer to the symbol name.
3. Specify the length of the symbol name and the channel number for the .ST file in AC1. (?SOPEN returns the channel number.)

The operating system returns the value of the specified symbol in AC0.

You must issue ?SCLOSE against the file with bit 0 of AC1 set. ?GTSVL issues ?SPAGES on the symbol table file, so you need to release the pages.

Notes

- See the descriptions of ?SOPEN, ?SCLOSE, and ?GTNAM in this chapter.

?GUNM

Gets the username of a process.

?GUNM
error return
normal return

Input

AC0 One of the following:

- −1 to obtain the username of the calling process
- PID of the target process
- Byte pointer to the name of the target process

AC1 One of the following:

- −1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID
- Any other value if AC0 contains −1

AC2 Byte pointer to the area that will receive the username (This area must be ?MXUN bytes long.)

Output

AC0 One of the following:

- 0 if the target process is not in Superuser mode
- 1 if the target process is in Superuser mode

AC1 Privilege word of the target process

AC2 Unchanged

Error Codes in AC0

ERMPR	System call parameter address error
ERPNM	Illegal process name format
ERPRH	Attempt to access a process not in hierarchy
ERVBP	Invalid byte pointer to name of target process passed in AC0

Licensed Material - Property of Data General Corporation

Why Use It?

?GUNM is useful if you know the simple process name or PID of a process, and you want to know the username associated with that process. (A full process name consists of the process's username and its simple process name.)

What It Does

?GUNM returns the target process's username terminated by a null to the location that you specify in AC2. Before you issue ?GUNM, load AC0 either with the PID of the target process or with −1 to obtain the caller's username. If you specify −1 in AC0, the operating system ignores AC1 because there is no need for a PID or byte pointer to the target's process name.

?GVPID

Gets the virtual PID of a process.

?GVPID
error return
normal return

Input

AC0	PID
AC1	Host ID in Bits 17 through 31
AC2	Reserved (Set to 0.)

Output

AC0	Virtual PID
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERHNE	Host does not exist (You supplied an invalid host ID in AC1.)
ERPRE	Illegal system call parameter (You supplied an invalid PID in AC0.)

Why Use It?

?GVPID encodes a host-ID/PID pair into a virtual PID for use in subsequent system calls.

What It Does

?GVPID returns a virtual PID. A virtual PID is a 16-bit value that references the PID (supplied in AC0) on the remote system (identified by the host ID in AC1). If the operating system has already assigned a virtual PID to the process, then ?GVPID returns the old virtual PID.

?HNAME — Gets a host name or host identifier.

?HNAME
error return
normal return

Case 1 — Given a host ID, return the host name
Case 2 — Given a host name, return the host ID
Case 3 — Return the local host ID and, optionally, the local hostname

Input

AC0 One of the following:

- Byte pointer to the buffer that you specify is to receive the host name (The buffer must be at least ?MXHN bytes long.) (Cases 1 and 3.)
- Byte pointer to the buffer that contains the host name (Case 2)
- 0 to return no local host name (Case 3)

AC1 One of the following:

- Host ID in Bits 17 through 31 (Case 1)
- 0 (Case 2)
- -1 (Case 3)

Output

AC0 Unchanged. (Case 1 —?HNAME returns the host name to the buffer that AC0 points to)

AC1 One of the following:

- Unchanged (Case 1)
- Host ID in Bits 17 through 31 (Case 2)
- Local host ID in Bits 17 through 31 (Case 3)

?HNAME Continued

AC2 Reserved (Set to 0.)

AC2 One of the following:

- Length (including the null terminator) of the host name that ?HNAME returned (Case 1)
- Undefined (Case 2)
- Unchanged (if input AC0 contains a 0) (Case 3)
- Length (including null terminator of local host name) (Case 3)

Error Codes in AC0

ERHID	Illegal host specification
ERHNE	Host does not exist
ERIHN	Illegal host name

Why Use It?

?HNAME allows you to determine host information by specifying the host ID or the host name.

What It Does

?HNAME lets you get a host name if you have the host ID or vice versa.

When you specify a host ID in Bits 17 through 31 of AC1, ?HNAME returns the host name to the buffer pointed to by AC0. When you specify a byte pointer to the host name in AC0, ?HNAME returns the host ID in Bits 17 through 31 of AC1.

However, ?HNAME also lets you get the local host ID and, optionally, the local host name. To do this, specify a byte pointer to the buffer that is to receive the local host name in AC0 (if you want the operating system to return the local host name) or 0 in AC0 (if you do not want the operating system to return the local host name) and -1 in AC1. Then, the operating system returns the local host ID in Bits 17 through 31 of AC1 and AC2 contains either a 0 (if AC0 contained 0 on input) or the length of the local host name, excluding the null terminator.

?IDEF

Defines a user device.

?IDEF
error return
normal return

Input

AC0 Contains the following:

- 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

- Bits 2 through 31 contain the device code for the user-defined device in the range from 1 through 191. (= 277 octal).

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

AC2 One of the following:

- Address of the map definition table
- Number of map slots needed (no map definition table) (each map slot accesses 1K words)

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Licensed Material - Property of Data General Corporation

?IDEF Continued

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

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.

Licensed Material - Property of Data General Corporation

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. The caller must be a resident process and must have privilege ?PVDV to use ?IDEF.

Before you issue ?IDEF, set up a device control table (DCT) in your logical address space and load its address into Bits 1 through 31 of AC1. (See Figure 2-69 for the DCT format if you want to issue ?IDEF from a 32-bit process. See Figure 2-70 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 second IOC (i.e., device codes 64.-127.), you must use PIO instructions to communicate with your device. NOVA I/O is limited to the first IOC.

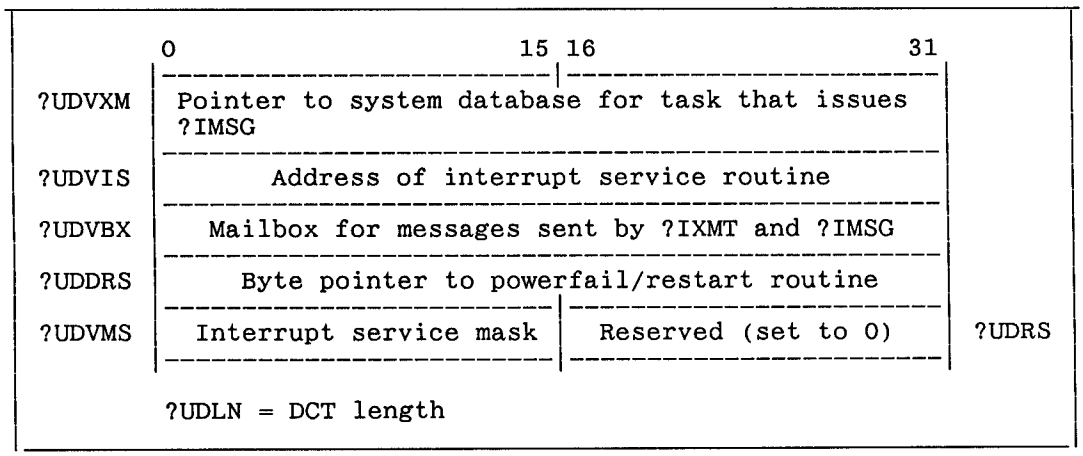


Figure 2-69. Structure of Device Control Table (DCT) for 32-Bit Processes

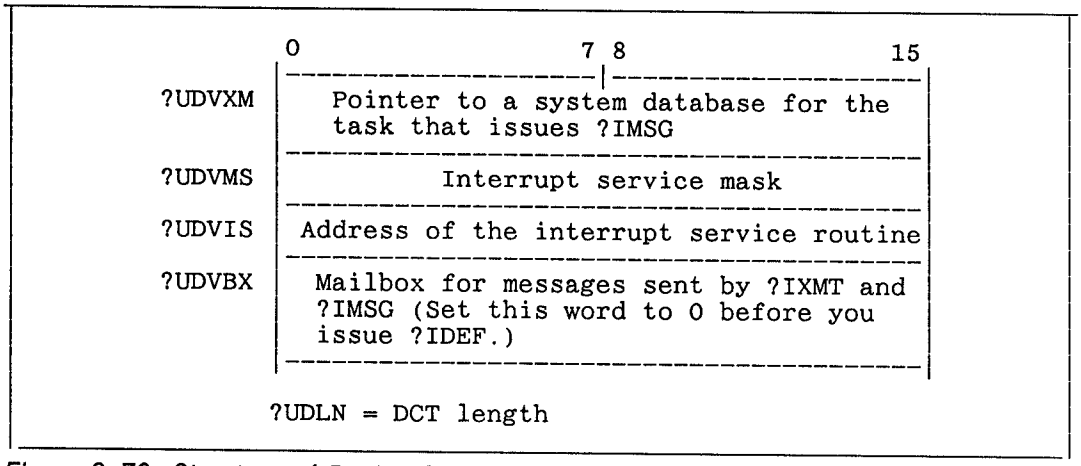


Figure 2-70. Structure of Device Control Table (DCT) for 16-Bit Processes

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.)

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 ?UDELTH words long. The entire table (with eight entries) is ?UDLTH words long. (See Figure 2-71 for the structure of a map definition table entry and see Table 2-48 for a description of its contents.)

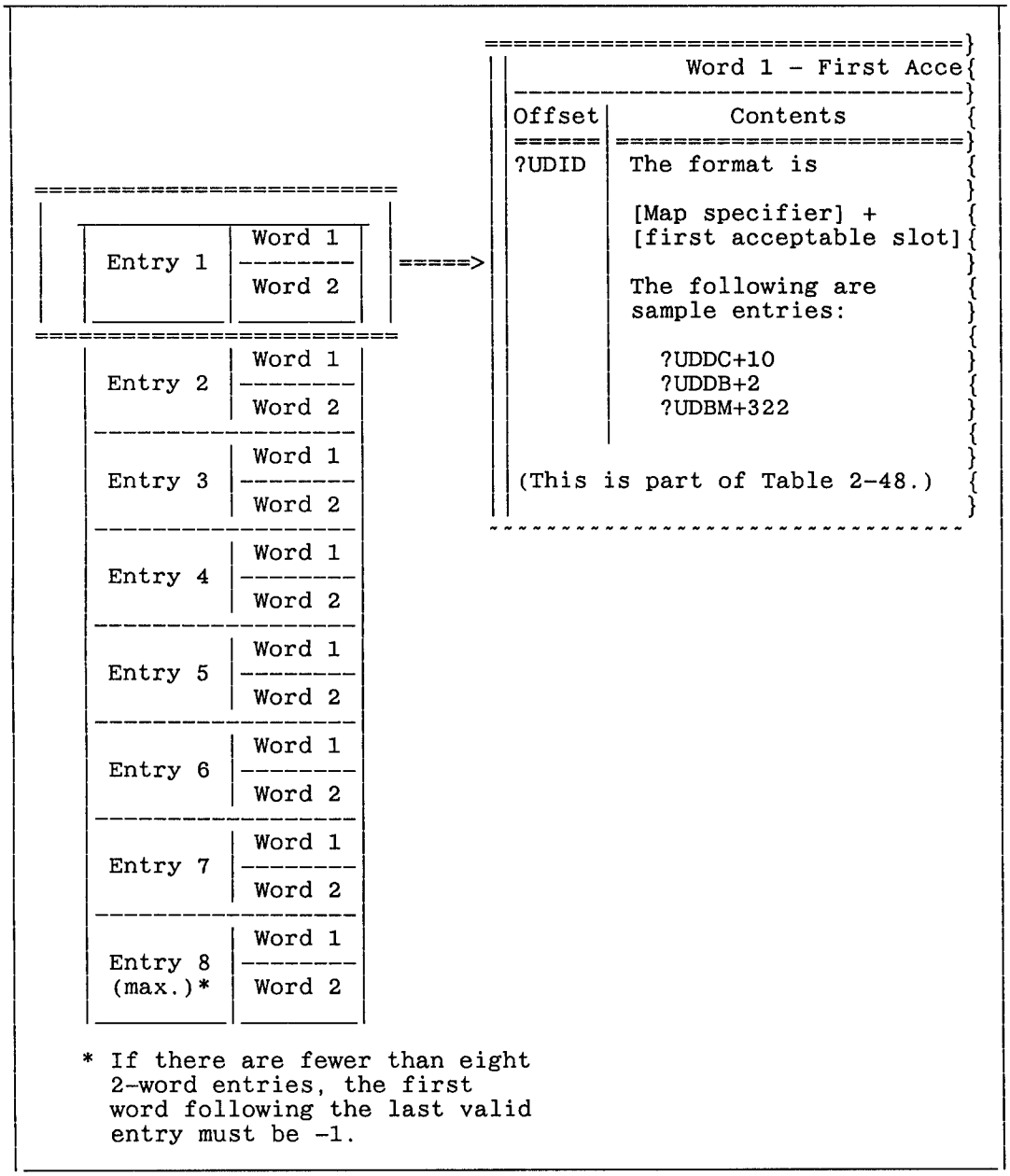


Figure 2-71. Structure of Map Definition Table

Table 2-48. Contents of Map Definition Table Entry

Word 1 – First Acceptable Map Slot		
Offset	Contents	Comments
?UDID	The format is [Map specifier] + [first acceptable slot] The following are sample entries: ?UDDC+10 ?UDDB+2 ?UDBM+322	Map specifier must be one of the following: – ?UDBM, which selects the BMC map. – ?UDDA, which selects the DCH A map. – ?UDDB, which selects the DCH B map. – ?UDDC, which selects the DCH C map. – ?UDDD, which selects the DCH D map. – ... – ?UDDP, which selects the DCH P map. First acceptable slot must be – From 0 through 1777 (octal), if your map specifier is ?UDBM. – From 0 through 37 (octal), if your map specifier is ?UDDA, ?UDDB, ?UDDC, or ?UDDD. The OS allocates the first contiguous group of slots on the map, starting with the first acceptable slot on the map that you selected. Then, the OS returns to you the first slot that it allocated in ?UDID.
Word 2 – Number of Map Slots Requested		
Offset	Contents	Comments
?UDNS	Number of map slots requested in range from 0 through 37 (octal).	The sum of the first acceptable slot plus the number of slots cannot be larger than the size of the map that you requested; that is, 37 (octal) for DCH or 2000 (octal) for BMC.

NOTE: If the OS cannot allocate all entries, then it does not allocate any entries.

Offset ?UDID, the map identifier word, defines which channel (the BMC or the DCH) the operating system should use for data transfers between the device and memory.

If you select data channel I/O, you can select one of the sixteen maps. The symbols ?UDDA through ?UDDP correspond to maps A through P. Add the correct map to the value corresponding to the first acceptable map slot you want the operating system to use. For example, the specification ?UDDA+5 tells the operating system to use map A (?UDDA) and to start with slot 5 in that map, if possible. Offset ?UDNS tells the operating system how many contiguous map slots you will need.

The map slots must be contiguous. Thus, if the first acceptable slot you specify is already in use, the operating system takes the next available map slot with the required number of contiguous slots.

Warning

If you issue ?IDEF system calls, you must make sure that all code and data touched by a user device interrupt handler service routine (including the handler itself) must be wired. Failure to do this will result in panics with code 14340 if a page fault occurs at interrupt level while trying to reference instructions or data.

Sample DCT

```
DCT:    .BLK    ?UDLN           ;Allocate enough space for the DCT
                                ;(DCT length = ?UDLN).
        .LOC    DCT+?UDVXM      ;TCB address of ?IMSG task.
        .DWORD  0               ;Set to 0. (The OS supplies this
                                ;value.)
        .LOC    DCT+?UDVIS      ;Interrupt service routine address.
        .DWORD  INTSRV          ;Interrupt service routine address is
                                ;INTSRV.
        .LOC    DCT+?UDVBX      ;?IXMT mailbox.
        .DWORD  0               ;Set to 0.
        .LOC    DCT+?UDDRS      ;Address of power-failure routine.
        .DWORD  -1              ;There is no power-failure routine.
        .LOC    DCT+?UDVMS      ;Interrupt service mask.
        .WORD   1B12+1B13+1B14+1B15
        .LOC    DCT+?UDRS       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    DCT+?UDLN       ;End of DCT.
```

Notes

- See the description of ?STMAP in this chapter.

?IDGOTO Redirects a task's execution path.

?IDGOTO
error return
normal return

Input

AC0	Address of the new code sequence
AC1	TID of the target task
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERTID	Task ID error (The OS could not find a matching TID.)
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?IDGOTO allows you to redirect a task that is currently executing, without having to kill the task and re-initiate it.

Because it will lift certain task suspensions, you can use ?IDGOTO to provide interrupt processing after a CTRL-C CTRL-A console interrupt. For example, you might create an interrupt task (via ?INTWT) to monitor your console for CTRL-C CTRL-A sequences. In this case, if the interrupt task were to detect an interrupt, it would issue ?IDGOTO to redirect the main task.

What It Does

?IDGOTO redirects the task that you specify in AC1 to a new code sequence, where it resumes executing the next time it gains CPU control. If the target task has any outstanding system calls, ?IDGOTO aborts those calls and readies the task. ?IDGOTO also readies the target task if it explicitly suspended itself with ?SUS, if it became suspended after issuing ?REC or ?XMTW, or if another task suspended it with ?IDSUS or ?PRSUS. The target task's priority remains the same after an ?IDGOTO.

Before you issue ?IDGOTO, load AC0 with the address of the new code sequence, and load AC1 with the TID of the target task. Note that a task may not use ?IDGOTO to redirect itself; thus, the TID that you specify in AC1 cannot be that of the calling task.

Notes

- See the descriptions of ?SUS, ?REC, ?XMTW, ?IDSUS, or ?PRSUS in this chapter.

?IDKIL — Kills a task specified by its TID.

?IDKIL
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	TID of the task you want to kill
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERTID	Task ID error (The OS could not find a matching TID.)

Why Use It?

Like the other system calls that require a task ID, ?IDKIL allows you to perform an action — in this case a "kill" — on a specific target task, rather than on all tasks of a given priority. Thus, you can use ?IDKIL to kill a task without affecting other tasks of the same priority.

What It Does

?IDKIL terminates the task that you specify in AC1. If the target task is suspended because of an outstanding system call, the operating system readies the target task by aborting the outstanding system call.

If you supplied a kill-processing routine for the target task with ?KILAD, the operating system passes control to that routine as it executes ?IDKIL. The kill-processing routine should end with a ?KILL system call. ?KILL, in turn, invokes a ?UKIL task-termination routine — either the system default ?UKIL or a ?UKIL you have defined. ?UKIL then terminates the task.

If you did not provide a kill-processing routine, control passes to the appropriate ?UKIL immediately.

Notes

- See the descriptions of ?KILAD, ?IDKIL, and ?KILL in this chapter.

Licensed Material - Property of Data General Corporation

?IDPRI — Changes the priority of a task specified by its TID.

?IDPRI
error return
normal return

Input

AC0	New priority for the target task
AC1	TID of the target task
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERTID	Task ID error (The OS could not find a matching TID.)

Why Use It?

?IDPRI is analogous to ?PRIPR, which changes the priority of a process. Both system calls allow you to influence the operating system scheduling activities by adjusting the priority numbers of a process or a task.

You can use ?IDPRI to reschedule tasks immediately. For example, you can issue ?IDPRI to give a task a higher relative priority than others for CPU control, or to give a task a lower priority (which upgrades the relative priorities of other tasks).

What It Does

?IDPRI changes the priority of the task that you specify in AC1. Before you issue ?IDPRI, load AC1 with the TID of the target task and load AC0 with the priority number you wish to assign. You can use ?IDPRI to change the priority of the calling task.

When you use ?IDPRI to assign a task a higher priority than the current ready task, the operating system reschedules the other task before it returns control to the ?IDPRI caller.

Notes

- See the descriptions of ?PRIPR and ?PRI in this chapter.

?IDRDY — Readies a task specified by its TID.

?IDRDY
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	TID of the target task
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERTID	Task ID error (The OS could not find a matching TID.)

Why Use It?

You can use ?IDRDY to revoke a task's suspension and, thus, ready the task for scheduling and eventual execution. Like the other tasking system calls that require a TID, ?IDRDY allows you to ready a single task, rather than all tasks of a given priority.

You can only issue system calls that require a TID against tasks that have nonzero TIDs. Thus, to ready a task without a TID, you must use ?PRRDY, not ?IDRDY.

What It Does

?IDRDY readies the task that you specify in AC1, but only if it was explicitly suspended by ?SUS, ?IDSUS, or ?PRSUS.

When you use ?IDRDY to ready a task that has a lower priority than the calling task, the operating system reschedules the tasks.

Notes

- See the descriptions of ?SUS, ?IDSUS, ?PRSUS, and ?PRRDY in this chapter.

?IDSTAT

Returns task status word. (16–Bit Processes Only)

```
?IDSTAT
error return
normal return
```

Input

AC0	Reserved (Set to 0.)
AC1	TID of the target task
AC2	Reserved (Set to 0.)

Output

AC0	Status word
AC1	Unchanged
AC2	Contains –1

Error Codes in AC0

ERTID	Task ID error (The OS cannot find the task whose TID you specified.)
ERICM	32–bit program incorrectly attempted to call ?IDSTAT

Why Use It?

?IDSTAT allows you to determine the status of a particular task.

What It Does

?IDSTAT returns a word that describes the target task's status; that is, the ?TSTAT word of the task's task control block (TCB).

Licensed Material – Property of Data General Corporation

?IDSUS Suspends a task specified by its TID.

?IDSUS
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	TID of the target task
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERTID	Task ID error (The OS could not find a matching TID.)

Why Use It?

Because ?IDSUS takes a TID as an input parameter, you can use it to explicitly suspend a single task, rather than all tasks of a given priority level. Also, a task can issue ?IDSUS to suspend itself.

What It Does

?IDSUS, which is the inverse of ?IDRDY, suspends the task that you specify in AC1. Note that you can only use ?IDSUS and the other system calls that require a TID against tasks that have a TID.

Notes

- See the description of ?IDRDY in this chapter.

?IESS Initializes an extended state save (ESS) area. (16–Bit Processes Only)

?IESS
normal return

Input

AC0	Size of the ESS area
AC1	Page 0 starting address
AC2	Starting address of the ESS area

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?IESS allows you to store any information that you consider relevant to a task.

What It Does

?IESS initializes an extended state save (ESS) area in the unshared portion of your logical address space. The ESS area is generally used to hold task–specific information for a 16–bit process, such as the value of the program counter and its carry bit, and the current contents of the accumulators. However, you can use it for other purposes.

Each task that uses the ESS area must issue ?IESS immediately upon entering the task code; otherwise, the ESS information may be invalid.

Input to ?IESS includes the starting address of the ESS area (in the unshared area of your logical address space), and a pointer to a block of Page 0 locations in your logical address space. When the operating system schedules a 16–bit task it copies the ESS information to the designated Page 0 area. When rescheduling occurs, the operating system transfers the ESS information back to the starting address you specified in AC2 when you first issued ?IESS.

?IFPU — Initializes the floating–point unit.

```
?IFPU
error return
normal return
```

Input

AC0	Starting address of an 18–word area designated to save the state of the floating–point status register
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?

Each task must issue ?IFPU before it issues floating–point instructions. This guarantees that every time the operating system gives control of the CPU to a task, the floating–point state is restored to the state the task was last in. If each task (even if your program only contains one task) does not issue ?IFPU first, your floating–point arithmetic results may not be correct.

What It Does

?IFPU reserves an 18–word area in which the operating system saves the state of the ECLIPSE MV/8000® floating–point status register.

Licensed Material - Property of Data General Corporation

?IHIST — Starts a histogram for a 16–bit process. (16–Bit Processes Only)

?IHIST
error return
normal return

Input

AC0	One of the following: • Byte pointer to the name of the target process • PID of the target process • –1 to start a histogram for the calling process	
AC1	One of the following: • –1 if AC0 contains a byte pointer • Any other value if AC0 contains –1 or a PID	
AC2	Address of the histogram packet	

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERHIS	Error on histogram initiation/deletion
ERMPR	System call parameter address error
ERPRH	Attempt to access process not in hierarchy
ERPRV	Caller not privileged for this action
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?IHIST allows a 16–bit resident process to monitor a range of addresses in its own or another process's logical address space. The operating system returns an array that compares the CPU time of the target process with the CPU time used by other processes, including the operating system. ?IHIST is a useful way of obtaining a global view of CPU activity.

What It Does

?IHIST, which is the 16-bit counterpart of ?WHIST, starts a histogram for a range of logical addresses in the target process. The calling process must be resident and cannot activate more than one histogram at a time.

Before you issue ?IHIST, load AC0 with the PID of the target process or a byte pointer to its name, or with -1 to start a histogram for the calling process. Your input to AC1 depends on your input to AC0.

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 2-72 shows the structure of the histogram packet for 16-bit processes.

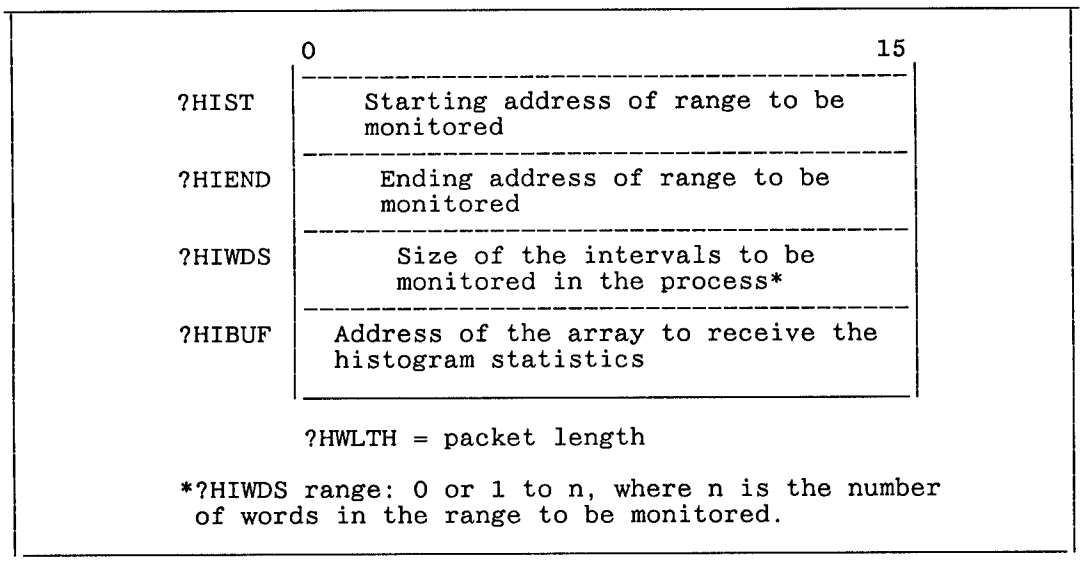


Figure 2-72. Structure of ?IHIST Packet

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 2-49 shows the structure of the histogram array for 16-bit processes.

Table 2-49. 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 ?HARAY+n*2-1	Total number of ticks in the nth interval.

?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

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.

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 console, 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.

?IMERGE

Modifies a ring field within a global port number.

?IMERGE
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 Ring field in Bits 29 through 31

Output

AC0 One of the following:

- Global port number, including the ring field (32-bit users only)
- OPH, which indicates high-order bits of global port number including the ring field (16-bit users only)

AC1 One of the following:

- Undefined (32-bit users only)
- OPL, which indicates low-order bits of global port number, including ring field (16-bit users only)

AC2 Unchanged

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?IMERGE allows both 16- and 32-bit users to modify the ring field within a global port so that it can receive an IPC message in an alternate ring.

What It Does

?IMERGE incorporates the new ring field that you specified in AC2 into the process's global port number.

Notes

- See the description of ?ISPLIT in this chapter.

?IMSG Receives an interrupt service message.

?IMSG
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	User DCT address of the sending device

Output

AC0	Undefined
AC1	Interrupt service message
AC2	Unchanged

Error Codes in AC0

ERMIM	Multiple ?IMSG calls to same DCT (Only one task can wait for an interrupt service message at a time)
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?IMSG is the "receive" counterpart of ?IXMT. ?IXMT and ?IMSG allow you to pass data from an interrupt service routine to an outside task, or to synchronize the two tasks.

Once a process has entered an interrupt service routine, ?IXMT and ?IXIT (exit from the routine) are the only system calls that process can issue.

What It Does

?IMSG lets the calling task receive a message sent by an interrupt service routine with ?IXMT. ?IXMT and ?IMSG can be in any order. If the receiving task issues the ?IMSG before the sender issues the ?IXMT, the operating system suspends the receiving task pending the transmission. As soon as the sending interrupt service routine issues ?IXMT, the receiver becomes active again, provided no other tasks suspended it in the meantime.

Before you can use ?IXMT and ?IMSG, you must use ?IDEF to define the sending interrupt service routine. Also, before you issue ?IMSG, you must load AC2 with the DCT address of the device that is associated with the sending interrupt service routine. When the operating system executes the sender's ?IXMT, it passes the message to the mailbox field that it has already set aside for the device. When the operating system executes ?IMSG, it moves the message from the mailbox to AC1.

You must define the user DCT address of the sending device, using ?IDEF, before you issue ?IMSG.

Notes

- See the descriptions of ?IXMT, ?IXIT, and ?IDEF in this chapter.

Licensed Material - Property of Data General Corporation

?INIT

Initializes a logical disk.

?INIT *[packet address]*
error return
normal return

Input

AC0	One of the following: • Byte pointer to a 32-byte receive buffer for the LD name • 0 if there is no buffer
AC1	One of the following: • -1 to graft the new LD to the system root directory (:) • 0 to graft the LD to the caller's working directory • Byte pointer to the pathname of the target directory
AC2	Address of the ?INIT packet, unless you specify the address as an argument to ?INIT

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?INIT packet

Error Codes in AC0

EREAD	Execute access denied
ERFIX	Can't initialize LD, run FIXUP on it
ERIDD	Inconsistent DIB (disk indentification block) data (Disk is not a valid operating system disk type.)
ERIDU	Incomplete LD (logical disk)
ERILD	Inconsistent LD
ERMIS	Disk and file system revision numbers don't match
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

Before you can use a logical disk, you must initialize it. You can do this with the CLI command INITIALIZE, or with the ?INIT system call. (You tell the operating system that the controller for the logical disk exists during the system-generation procedure.) Initializing a logical disk allows you to incorporate its files into the existing file structure.

What It Does

?INIT initializes a logical disk and grafts the LD's local directory structure to the directory that you specify in AC1 (that is, the system root, the current working directory, or another directory).

To use ?INIT, the calling process must have Owner access to the target LD's root and either Write or Append access to the target directory. In addition, the caller must have Execute access to each disk unit in the LD.

?INIT requires a packet of 2n + 1 words, where n is the number of disk units that make up the logical disk you are initializing. Table 2-50 lists the contents of the ?INIT packet.

Table 2-50. Contents of ?INIT Packet*

Offset	Contents
0	Number of physical units in the LD.
1 and 2 (double word)	Byte pointer to the name of the first disk in the LD.
3 and 4 (double word)	Byte pointer to the name of the second disk in the LD.
5 and 6 (double word)	Byte pointer to the name of the third disk in the LD.
...	
2*n-1 and 2*n (double word)	Byte pointer to the name of the nth disk in the LD.

* There is no default unless otherwise specified.

You must terminate each disk name with a null.

Notes

- Refer to your operating system's "How to Generate and Run" manual for a list of the valid disk names and for information on the system-generation procedure.

?INTWT — Defines a console interrupt task.

?INTWT
error return
normal return

Input

None

Output

None

Error Codes in AC0

No error codes are currently defined.

Why Use It?

The operating system enables console interrupts when a program begins to execute. Before the operating system will recognize a console interrupt, however, you must define a console-interrupt task with ?INTWT. Because the operating system disables console interrupts after it executes the interrupt task, you must issue another ?INTWT each time you want to re-enable console interrupts.

What It Does

?INTWT defines a task to handle a console interrupt (that is, a CTRL-C CTRL-A control sequence, which suspends the process console's output). In addition, ?INTWT re-enables console interrupts if they were previously disabled by a console interrupt.

When the operating system detects a console interrupt, it passes control to the task defined by ?INTWT, and then disables further console interrupts until you issue either another ?INTWT or a ?OEBL. You should initiate the console-interrupt task at the highest priority so that it will receive control as soon as possible to service the interrupt.

Notes

- See the description of ?OEBL in this chapter.

?IQTSK — Creates a queued task manager.

?IQTSK
error return
normal return

Input

AC0	Priority level for the queued task manager (0 to assign it the caller's priority level)
AC1	TID for the queued task manager
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERQTS	Error in queued task request (An active queued task manager already exists.)

Why Use It?

You must issue ?IQTSK before you can issue ?TASK to create queued tasks. Queued tasks allow your program to delay task initiation and/or direct the operating system to reinitiate tasks at specific time intervals.

What It Does

?IQTSK creates a queued task manager task to manage tasks you initiate with the ?TASK queued task creation facility. You can create only one queued task manager. Therefore, you can issue only one ?IQTSK per program. However, when you issue ?IQTSK, it creates a task. Therefore, you must allow for one extra task control block (TCB). For example, if you create three tasks with one ?TASK system call, which you want queued, the operating system is using five TCBs.

Before you issue ?IQTSK, load AC0 with a priority level for the queued task manager, and load AC1 with its task identifier (TID). You must specify a unique task ID, because the operating system needs this information to control the task manager. If you specify 0 in AC0, the operating system assigns the queued task manager the same priority as the calling task.

When you terminate the last active task in a process, the operating system terminates the queued task manager, provided it is still active and has an empty task initiation queue.

Notes

- See the descriptions of ?TASK and ?DQTASK in this chapter.

?IREC

Receives an IPC message.

?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 message.

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 2-73 shows the header's structure, and Table 2-51 describes its contents. Table 2-51 also describes the optional contents of ?ISFL in the ?IREC header.

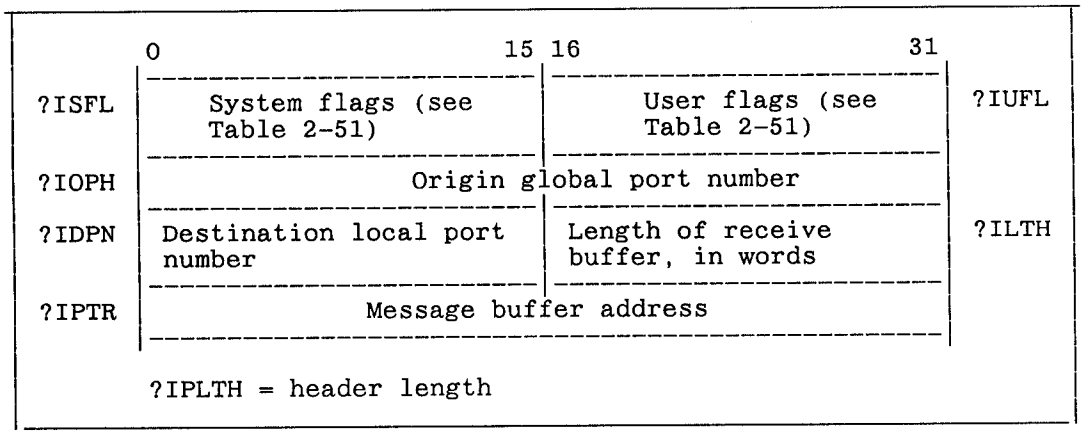


Figure 2-73. Structure of ?IREC Header

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 2-51 which describes the system flag words and the 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.

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

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 process termination 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 message, it must issue an ?IREC system call with offset ?IOPH set to a special system port, ?SPTM.

Table 2-51. 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 2-74.)
?IOPH (double word)	Origin global port number.
?IDPN	Destination local port number.
?ILTH	Size of the receive buffer, in words.
?IPTR (double word)	Address of the receive buffer.

* There is no default unless otherwise specified.

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 2-74 shows the structure of offset ?IUFL.

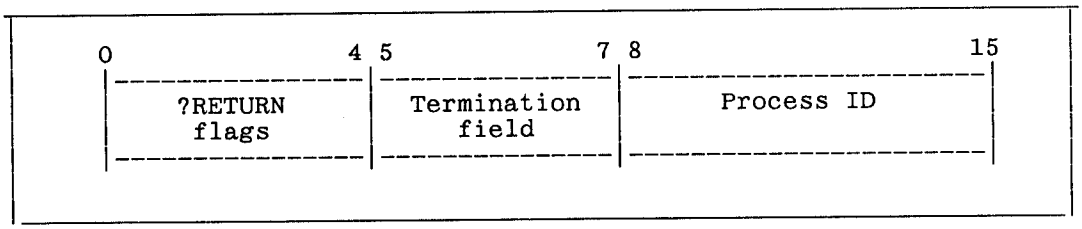


Figure 2-74. Structure of Offset ?IUFL

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 2-53, 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 2-51 for the ?RETURN flags.

Table 2-52. Termination Codes for 16-Bit Processes

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.

Licensed Material - Property of Data General Corporation

Table 2-53. Process Termination Codes in Offset ?IUFL for ?IREC and ?ISEND Headers

Code	Meaning
?TSELF	Either a 16-bit process terminated itself with a ?TERM or a ?RETURN system call or a 32-bit process terminated itself with a ?TERM system call.
?TRAP	A user trap terminated a 16-bit process; Word 5 of the IPC message to the father describes the trap.
?TCIN	An abort console interrupt (CTRL-C CTRL-B sequence) terminated a process.
?TSUP	Terminated by a superior process.
?TAOS	The O/S terminated a process because of an error; offset ?IPTR in the IPC header contains the error code.
?TBCX	A process broke a connection that was established via the connection-management system calls.
?TCCX	The connection still exists, but the process chained.
?TEXT	Indicates an extended termination code; the extended code appears in offset 0 (first word) of the IPC message. A termination code of ?TEXT means that the actual termination code is a 16-bit code in the first word of the termination message in the receive buffer. The The following list describes these extended termination codes.

Extended Code	Meaning
?T32T	A 32-bit process terminated itself with a ?TERM or a ?RETURN system call.
?TR32	A 32-bit process terminated because of a user trap; word 10 of the termination message describes the trap.
?TABR	Task abort notification to a server process. This involves customer/server relationship, ?IDGOTO, ?IS.R, and ?IREC, either 16-bit or 32-bit process.

Licensed Material - Property of Data General Corporation

If the message is in standard CLI format, the return flag is set to ?RFCF, and the remaining flags (?RFEC, ?RFWA, ?RFER, 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.

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 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 2-52 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 2-54.

If the process terminated because of an abort console 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 2-54. ?TRAP Termination Messages for 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: 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.

Termination Messages for 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 2-55. The length of this message is ?TPLN words.

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.
```

Table 2-55. ?TEXT Code Termination Messages Sent on 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.
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.)

(Continues)

Table 2-55. ?TEXT Code Termination Messages Sent on 32-Bit Process User Trap

Word	Contents	
11 (Cont.)	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)

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.

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. To issue ?IRMV, a process must have privilege ?PVDV.

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.

?ISEND — Sends an IPC message.

?ISEND *[header address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?ISEND header

Error Codes in AC0

ERIDP	Illegal destination port
ERPRV	Caller not privileged for this action
ERVWP	Invalid word pointer passed as a system call argument
ERSPC	File space exhausted (The IPC spool file is full.)

Why Use It?

?ISEND allows you to pass freeform messages from your current process to a specific receiving process. The ?ISEND and ?IREC system calls are useful for passing variables and other data from one process to another and for synchronizing two processes.

What It Does

?ISEND opens an IPC port for the calling process and sends the message that you specify in the ?ISEND header to a designated receiver. To issue ?ISEND, a process must have privilege ?PVIP or be sending a message to a customer.

Before you issue ?ISEND, define the message and the ?ISEND header in your logical address space. You can either cite the header address as an argument to ?ISEND or you can load the address into AC2 before you issue ?ISEND. The ?ISEND header consists of ?IPLTH words. Figure 2–75 shows the ?ISEND header's structure and Table 2–56 describes its contents.

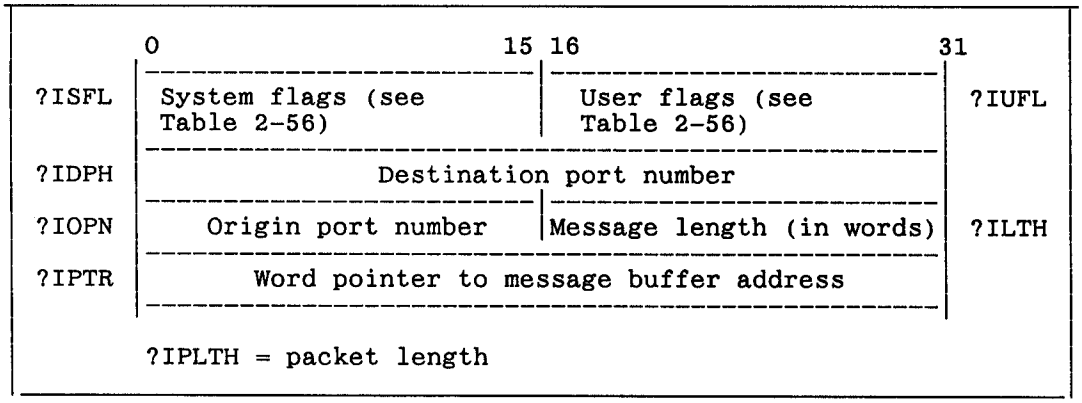


Figure 2-75. Structure of ?ISEND Header

Table 2-56. Contents of ?ISEND Header*

Offset	Contents
?ISFL	System flag word ?IFSTM--Loop the message. (Send the message back to the sender.) ?IFNSP--Do not spool the message; signal an error if there is no ready receiver.
?IUFL	User flag word. (See Figure 2-74.)
?IDPH (double word)	Destination port number.
?IOPN	Origin port number.
?ILTH	Message buffer length, in words.
?IPTR (double word)	Address of the message buffer.

* There is no default unless otherwise specified.

Offset ?IDPH specifies the destination's global port number. (The ?ILKUP and ?TPORT system calls return this information.) Set offset ?IOPN (the origin port) to the local port number you want to use for this ?ISEND.

Set offset ?IPTR to the address of the message in your logical address space. Specify the length of the message buffer (that is, the length of the message) in offset ?ILTH. If ?ILTH is zero, the operating system sends the message and does not verify ?IPTR. If ?ILTH is nonzero, the operating system sends the message and also verifies ?IPTR.

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.

?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.

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.

?IS.R Sends and then receives an IPC message.

?IS.R *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?IS.R header

Error Codes in AC0

ERIDP	Illegal destination port
ERMPR	System call parameter address error (The receive buffer is not in the unshared area of your address space.)
ERPRV	Caller not privileged for this action
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?IS.R allows you to send a message to a receiver and wait for a reply on the same port. ?IS.R performs a special case of ?ISEND followed by an ?IREC with a slightly lower overhead.

What It Does

?IS.R performs an ?ISEND and, if successful, follows it immediately with an ?IREC. To issue ?IS.R, a process must have privilege ?PVIP or have a connection to the target process.

Before you issue ?IS.R, you must define the message and set up a receive buffer (for the receiver's reply) in your logical address space. In addition, you must set up the ?IS.R packet, a combination of the ?ISEND and ?IREC packets that consists of ?IPRLTH words. You can cite the packet address as an argument to ?IS.R or load its address into AC2 before you issue ?IS.R. Figure 2-76 shows the structure of the ?IS.R packet, and Table 2-57 describes its contents.

?IS.R Continued

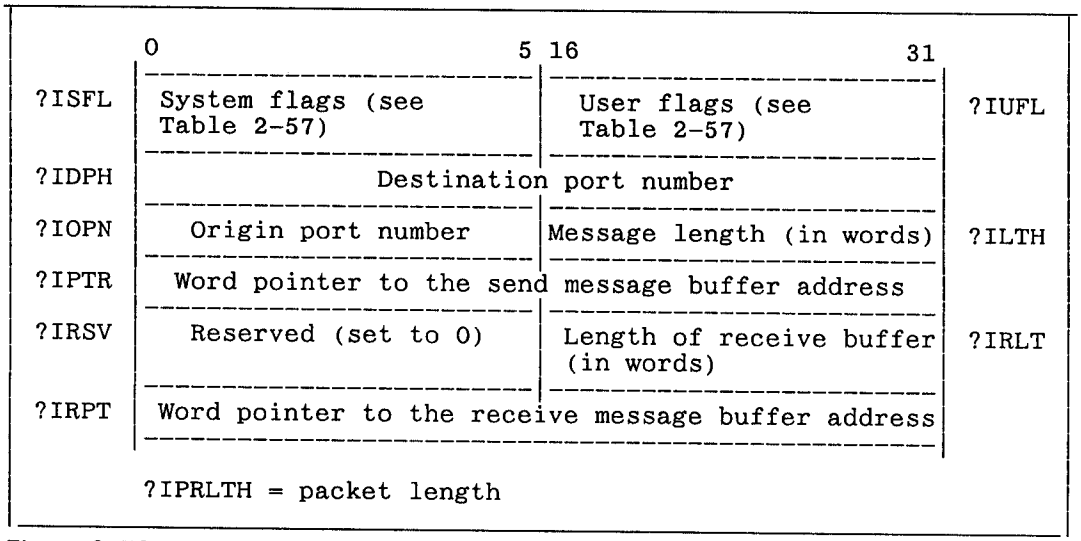


Figure 2-76. Structure of ?IS.R Header

The message is sent and the reply is received, but not necessarily with the same port numbers. (The ?TPORT system call translates a local port number to its global equivalent.)

During message transmission, the operating system suspends the ?IS.R calling task until it receives the reply or until it takes an error return.

Table 2-57. Contents of ?IS.R Header*

Offset	Contents
?ISFL	System flag word ?IFSTM--Loop the message. (Send the message back to the sender.) ?IFNSP--Do not spool the message; signal an error if there is no ready receiver. ?IFRFM--Receive a looped message (sent by this process to itself). ?IFSOV--Spool the message if the receive buffer is too small. ?IFBNK--Signal an error if there is no spooled message for this receiver. ?IFRING--Contains the sender's ring field (returned by the OS). (This is a 3-bit quantity.) ?IFPR---Indicates .PR file type of sender; 0 if sender is a 32-bit process; 1 if sender is a 16-bit process (returned by the OS).
?IUFL	User flag word. (See Figure 2-74.)
?IDPH (double word)	Destination port number.
?IOPN	Origin port number.
?ILTH	Length of the message buffer, in words.
?IPTR (double word)	Word pointer to the address of the message buffer.
?IRSV	Reserved. (Set to 0.)
?IRLT	Length of the receive buffer, in words.
?IRPT (double word)	Word pointer to the address of the receive buffer.

* There is no default unless otherwise specified.

Sample Header

```
HDR:    .BLK    ?IPRLTH         ;Allocate enough space for the packet
                                ;Packet length = ?IPRLTH.
        .LOC    HDR+?ISFL       ;System flags.
        .WORD   0               ;None.
        .LOC    HDR+?IUFL       ;User flags.
        .WORD   0               ;Set to 0.
        .LOC    HDR+?IDPH       ;Global port no. of receiving process.
        .DWORD  5               ;(?TPORT returns this information.)
        .LOC    HDR+?IOPN       ;Origin port no.
        .WORD   7               ;Local port no. to use for this ?IS.R.
        .LOC    HDR+?ILTH       ;Message buffer length (in words).
        .WORD   0               ;Set buffer length before each ?IS.R.
        .LOC    HDR+?IPTR       ;Message buffer address (must be in
        .DWORD  BUFF*2          ;unshared area of your logical address
                                ;space).  Byte pointer to BUFF is
                                ;message buffer address.
        .LOC    HDR+?IRSV       ;Reserved
        .WORD   0               ;You must set this value to 0.
        .LOC    HDR+?IRLT       ;Receive buffer length (in words).
        .WORD   0
        .LOC    HDR+?IRPT       ;Receive buffer address.
        .DWORD  -1
        .LOC    HDR+?IPLTH      ;End of ?ISEND header.
```

Notes

- See the descriptions of ?ISEND, ?IREC, and ?TPORT in this chapter.

?ITIME — Returns the OS–format internal time.

?ITIME
error return
normal return

Input

None

Output

AC0	Undefined
AC1	Time in 32,768ths of a second
AC2	Days since 1 January 1968

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?ITIME provides you with a time stamp.

What It Does

?ITIME returns the current date and time in AC1 and AC2. However, although ?ITIME returns the time to 32,768ths of a second, it is only as accurate as the real–time clock. To obtain the frequency — and therefore the accuracy — of the real–time clock, issue the ?GHRZ system call.

?IXIT Exits from an interrupt service routine.

?IXIT
no error return
no normal return

Input

AC0	Reserved (Set to 0.)
AC1	One of the following: • 0 to suppress task scheduling upon return • Any nonzero value, to re-enable task scheduling
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

None (There is no error return.)

Why Use It?

You must issue ?IXIT if you want to return the calling process from an interrupt service routine. Once a process has entered such a routine, ?IXIT and ?IXMT (which transmits an interrupt message) are the only system calls that process can issue. You cannot issue these system calls from any other kinds of routines.

What It Does

?IXIT returns the calling process from an interrupt service routine. (Control passes to the operating system after this system call.) By loading AC1 with 0, you can direct the operating system to suppress task scheduling upon return from the interrupt service routine.

Notes

- See the description of ?IXMT in this chapter.

?IXMT

Transmits a message from an interrupt service routine.

?IXMT
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Message
AC2	DCT address of the sending device

Output

AC0	Undefined
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERXMT	Signal to address already in use (that is, the mailbox)
ERXMZ	Attempt to XMT illegal message

Why Use It?

?IXMT and ?IMSG allow you to send data from an interrupt service routine to an outside task, or to synchronize the routine with an outside task.

What It Does

?IXMT sends a message up to 32 bits long from an interrupt service routine to a specific receiving task outside the sending routine. (The receiving task issues ?IMSG to receive the message.) ?IXMT and ?IXIT are the only system calls you can issue from an interrupt service routine. In addition, you cannot issue these system calls from any other kind of routine.

Before you issue ?IXMT, load AC2 with the user DCT address of the device associated with the sending routine, and load AC1 with the message. The message must be nonzero, or ?IXMT fails on error ERXMZ.

When the operating system executes ?IXMT, it passes the message to the mailbox that it associates with the device. (Be sure to initialize the mailbox before you issue ?IXMT.) Then, when the receiving task issues the complementary ?IMSG, the operating system passes the message from the mailbox to AC1. If the mailbox already contains a nonzero value, the ?IXMT fails on error code ERXMT.

If a sending routine issues ?IXMT before the receiver issues ?IMSG, the operating system holds the message in the mailbox for later delivery. If the ?IMSG occurs before the ?IXMT, the operating system suspends the receiving task until the transmission occurs.

Notes

- See the descriptions of ?IXIT and ?IMSG in this chapter.

?JPINIT — Initializes a job processor.

AOS/VS only

?JPINIT *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?JPINIT packet

Error Codes in AC0

ERIJP	Invalid JPID
ERILP	LPID does not exist
ERJAI	JP already initialized
ERMFF	Microcode format error
ERNML	No microcode loaded
ERNMP	Not a multiprocessor system
ERPRV	Caller not privileged for this action
ERVBP	Invalid byte pointer passed as system call argument

Why Use It?

Use this system call to initialize a job processor (JP) that is not currently active.

What It Does

This system call initializes an inactive job processor. AOS/VS attaches the JP to the specified logical processor (LPID in the parameter packet) and begins to schedule processes to run on the JP.

AOS/VS selects microcode for the JP in one of three ways according to the values of two bits in word JPI_PKT.FLAGS of the parameter packet in Figure 2-77.

First, if you set bit ?JPI_PKT.FLAGS_IN_UC (and provide a byte pointer in double word ?JPI_PKT.UC_NAME), AOS/VS loads the specified microcode file into the JP. AOS/VS also requires bit ?JPI_PKT.FLAGS_EX_UC to be zero.

Second, if you do not set bit ?JPI_PKT.FLAGS_IN_UC, AOS/VS looks at bit ?JPI_PKT.FLAGS_EX_UC. A value of 1 tells AOS/VS NOT to load microcode into the JP; instead, AOS/VS uses the microcode that is currently in the JP.

Third, if you don't set bits ?JPI_PKT.FLAGS_IN_UC and ?JPI_PKT.FLAGS_EX_UC (i.e., if you don't provide a microcode pathname and don't want to use existing microcode), AOS/VS loads a standard microcode file based on the CPUID of the JP.

Suppose an error occurs while ?JPINIT loads a microcode file. The OP console does NOT receive any messages. However, AOS/VS returns an error code to your program.

Figure 2-77 shows the structure of the ?JPINIT parameter packet, and Table 2-58 describes its contents.

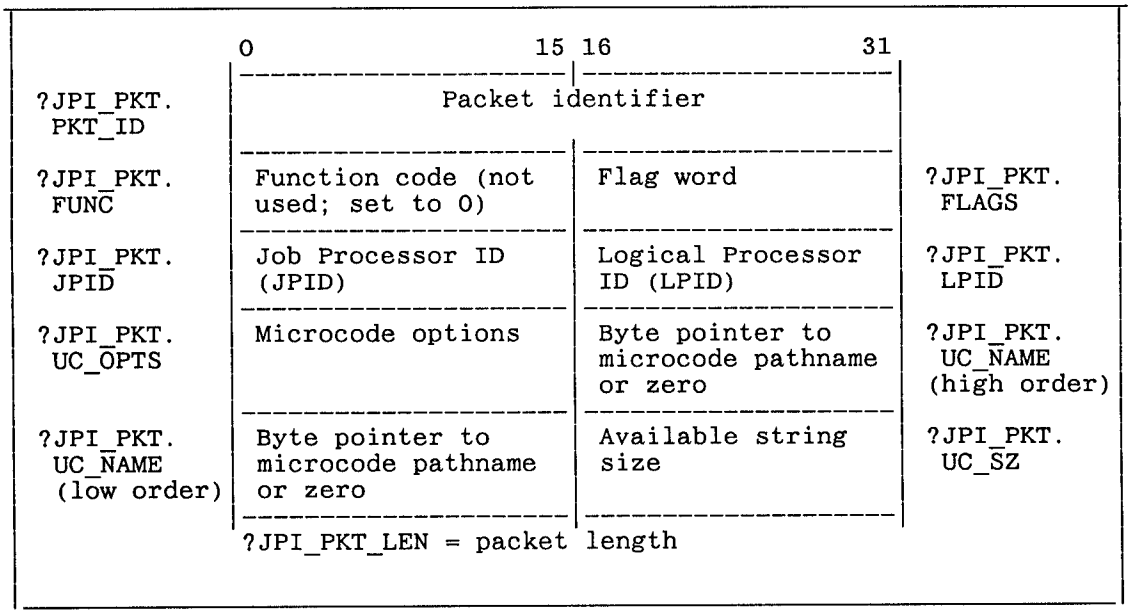


Figure 2-77. Structure of ?JPINIT Packet

?JPINIT Continued

Table 2-58. Contents of ?JPINIT Packet

Offset	Contents
?JPI_PKT.PKT_ID (double word)	Packet identifier. Place ?JPI_PKT_PKTID here.
?JPI_PKT.FUNC	Function code. Not used. (Set to 0.)
?JPI_PKT.FLAGS	Flag word to control microcode file loading. Legal values of bit pairs ?JPI_PKT.FLAGS_IN_UC and ?JPI_PKT.FLAGS_EX_UC are 10, 01, and 00.
?JPI_PKT.JPID	Job processor ID. It must be between values ?JPID_MIN and ?JPID_MAX, inclusive.
?JPI_PKT.LPID	Logical processor ID. It must be between values ?LPID_MIN and ?LPID_MAX, inclusive.
?JPI_PKT.UC_OPTS	Microcode options word. This offset corresponds directly with bits 0-15 of AC0 for the JPLCS instruction. The options in this offset are specific to the processor and cannot be listed here. However, if this offset contains zero, you have specified all the default options.
?JPI_PKT.UC_NAME (double word)	Byte pointer to the microcode file's pathname, or 0 if you are not specifying a pathname.
?JPI_PKT.UC_SZ	Number of bytes in the buffer containing the microcode file's pathname, or 0 if you are not specifying a pathname.

?JPMOV

Moves a job processor to a new logical processor.

AOS/VS only

?JPMOV *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?JPMOV packet

Error Codes in AC0

ERIJP	Invalid JPID
ERILP	Invalid LPID
ERJNI	JP is not initialized
ERLNE	LP does not exist
ERLJP	Attempt to release last JP attached to an LP
ERJAA	JP already attached to LP
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to move a job processor (JP) to another logical processor (LP).

What It Does

This system call moves a job processor, if it is attached to an existing logical processor, to the specified logical processor. After this call executes successfully, AOS/VS schedules jobs on the JP according to the new LP's class designations.

When you move a JP, it's possible that you could leave an LP behind that has no JP attached to it. To prevent this from happening, set a bit in the flag word of the parameter packet. Then, you will get an error if you leave an "unattached" LP. If you do not set this bit, AOS/VS will move the JP regardless of whether or not it would result in an LP without a JP.

?JPMOV Continued

Figure 2-78 shows the structure of the ?JPMOV parameter packet, and Table 2-59 describes its contents.

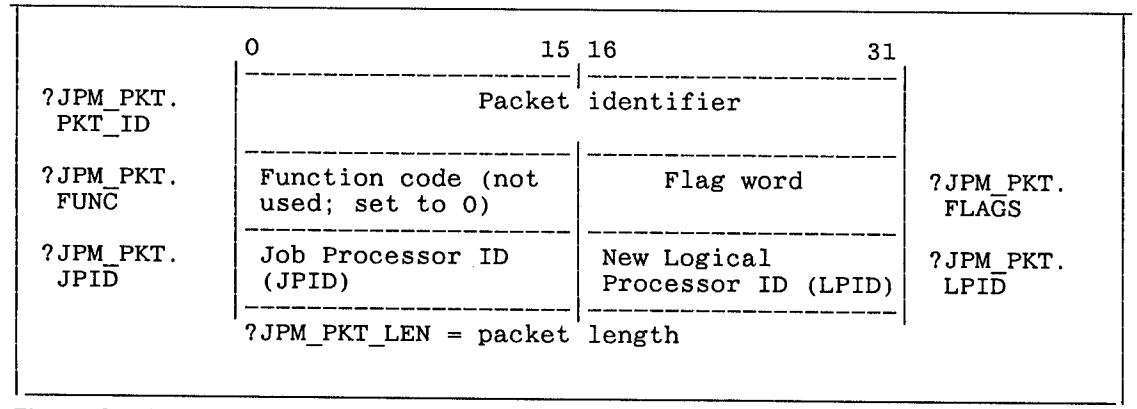


Figure 2-78. Structure of ?JPMOV Packet

Table 2-59. Contents of ?JPMOV Packet

Offset	Contents
?JPM_PKT.PKT_ID (double word)	Packet identifier. Place ?JPM_PKT_PKTID here.
?JPM_PKT.FUNC	Function code. Not used. (Set to 0.)
?JPM_PKT.FLAGS	Flag word. If bit ?JPM_PKT.FLAGS_LJPID is set, AOS/VS returns error code ERLJP when you try to move a JP that would leave an unattached LP; the JP remains attached to the LP. If the bit is not set, AOS/VS returns no error code under the same circumstances; the JP is removed from the old LP and attached to the new LP.
?JPM_PKT.JPID	Job processor ID. It must be between values ?JPID_MIN and ?JPID_MAX, respectively.
?JPM_PKT.LPID	New logical processor ID. It must be between values ?LPID_MIN and ?LPID_MAX, respectively.

?JPREL — Releases a job processor.

AOS/VS only

?JPREL *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?JPREL packet

Error Codes in AC0

ERIJP	Invalid JPID
ERJNI	JP is not initialized
ERJST	JP running one or more system tasks
ERLJP	Attempt to release last JP attached to an LP
ERNMP	Not a multiprocessor system
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to release a job processor (JP) from your AOS/VS system.

What It Does

This system call releases a job processor.

AOS/VS returns an error if the JP is not initialized. AOS/VS also returns an error if the JP is performing a vital system task such as file I/O.

If the JP is initialized and not performing a vital system task, AOS/VS removes the JP from its associated logical processor (LP) and stops the JP from processing. When you release a JP, it's possible that you could leave an LP behind that has no JP attached to it. To prevent this from happening, set a bit in the flag word of the parameter packet. Then, you will get an error if you leave an "unattached" LP. If you do not set this bit, AOS/VS will remove the JP regardless of whether or not it would result in an LP without a JP.

Licensed Material – Property of Data General Corporation

?JPREL Continued

Figure 2-79 shows the structure of the ?JPREL parameter packet, and Table 2-60 describes its contents.

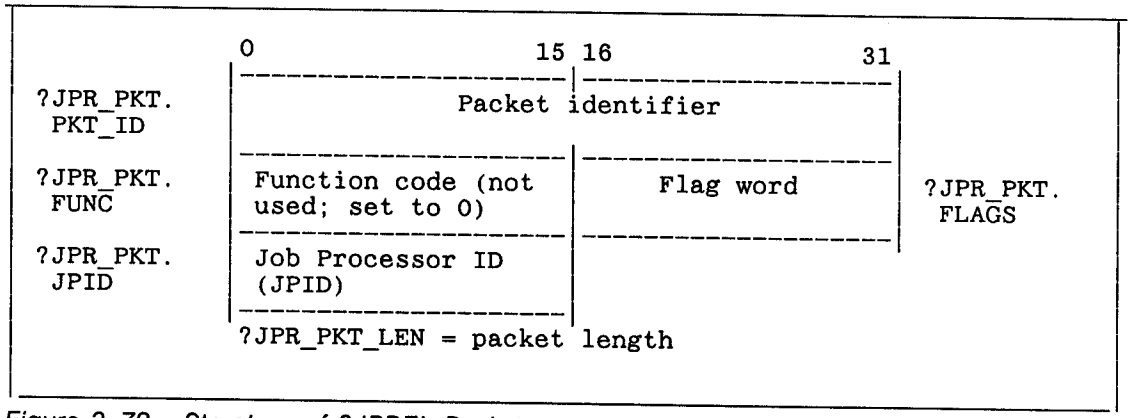


Figure 2-79. Structure of ?JPREL Packet

Table 2-60. Contents of ?JPREL Packet

Offset	Contents
?JPR_PKT.PKT_ID (double word)	Packet identifier. Place ?JPR_PKT_PKTID here.
?JPR_PKT.FUNC	Function code. Not used. (Set to 0.)
?JPR_PKT.FLAGS	Flag word. If bit ?JPR_PKT.FLAGS_LJPID is set, AOS/VS returns error code ERLJP when you try to release a JP that would leave an unattached LP; the JP remains attached to the LP. If the bit is not set, AOS/VS returns no error code under the same circumstances; the JP is removed from the LP.
?JPR_PKT.JPID	Job processor ID. It must be between values ?JPID_MIN and ?JPID_MAX, respectively.

Licensed Material - Property of Data General Corporation

?JPSTAT — Gets the status of a job processor.

AOS/VS only

?JPSTAT *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?JPSTAT packet

Error Codes in AC0

ERIJP	Invalid JPID
ERJNI	JP is not initialized
ERICD	Invalid function code
ERNMP	Not a multiprocessor system

Why Use It?

Use this system call to obtain information about all initialized job processors (JP) or about a specific initialized job processor.

What It Does

This system call has two versions — general and specific. The general version returns the total number and JPIDs of all initialized job processors. The specific version returns status information of a specific initialized job processor. You tell AOS/VS which version you want by placing a code in offset ?JPS_PKT.FUNC of the main parameter packet.

If you select the general version, then AOS/VS returns, in a double word of the general status subpacket, a bit map of initialized job processors that operates as follows:

- If a bit is set, then the JP with the corresponding JPID is initialized.
- If a bit is not set, then the JP with the corresponding JPID is not initialized.

For example, suppose bits 2, 3, and 5 are set. This means JPs with JPIDs of 2, 3, and 5 are initialized. Furthermore, JPs with JPIDs of 0, 1, 4, 6, 7, 8, ..., 15 are not initialized.

?JPSTAT Continued

If you select the specific version, then AOS/VS returns, in the specific status subpacket, the following information for the target JP:

- Information that the assembly language instruction JPSTATUS returns. This information is in the five words beginning with ?JPS_SPEC.STATE.
- An indication, in the software flags word, whether the JP is the "mother" processor and whether the JP is running a system task. In either case, you cannot release the JP.
- Which LP the JP is attached to.

Figure 2-80 shows the structure of the ?JPSTAT main parameter packet, and Table 2-61 describes its contents. Figure 2-81 and Figure 2-82 show the structure of the general status subpacket and specific status subpacket, respectively. Table 2-62 and Table 2-63 describe the contents of these respective subpackets.

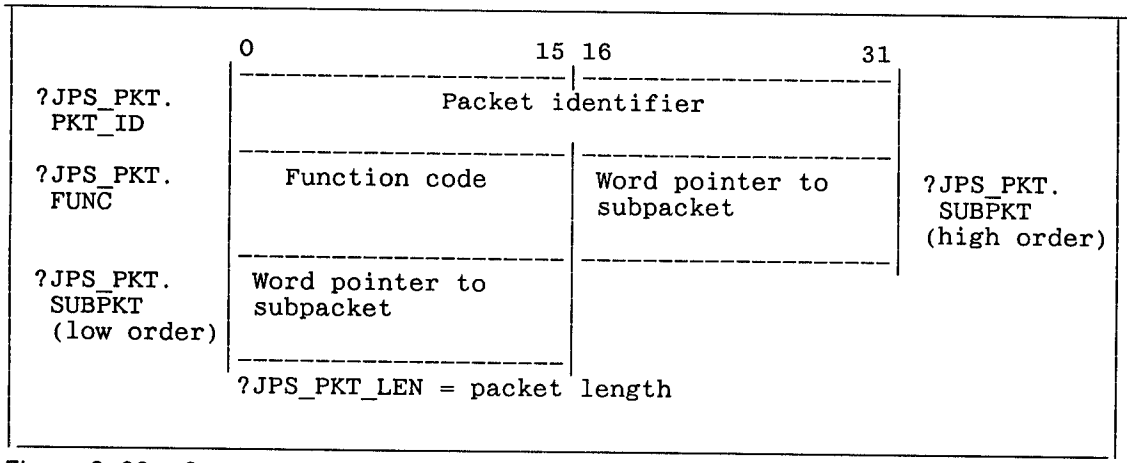


Figure 2-80. Structure of ?JPSTAT Main Packet

Table 2-61. Contents of ?JPSTAT Main Packet

Offset	Contents
?JPS_PKT.PKT_ID (double word)	Packet identifier. Place ?JPS_PKT_PKTID here.
?JPS_PKT.FUNC	General/Specific code. Place ?JPS_GEN here to obtain general information or ?JPS_SPEC to obtain specific information.
?JPS_PKT.SUBPKT (double word)	If you have selected general information, place the word address of the general status subpacket here. If you have selected specific information, place the word address of the specific status subpacket here.

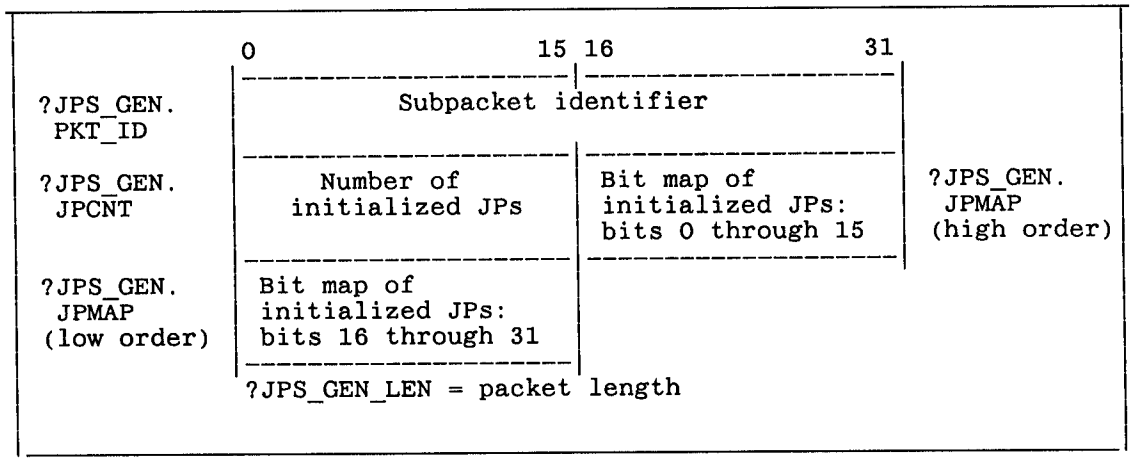


Figure 2-81. Structure of ?JPSTAT General Information Subpacket

Table 2-62. Contents of ?JPSTAT General Information Subpacket

Offset	Contents
?JPS_GEN.PKT_ID (double word)	Subpacket identifier. Place ?JPS_GEN_PKTID here.
?JPS_GEN.JPCNT	Number of job processors that are currently initialized.
?JPS_GEN.JPMAP (double word)	Bit map to indicate the JPIDs of the job processors that are currently initialized. For example, if AOS/VS has set the leftmost bit of the first (high order) word in this double word, then job processor number 0 is currently initialized.

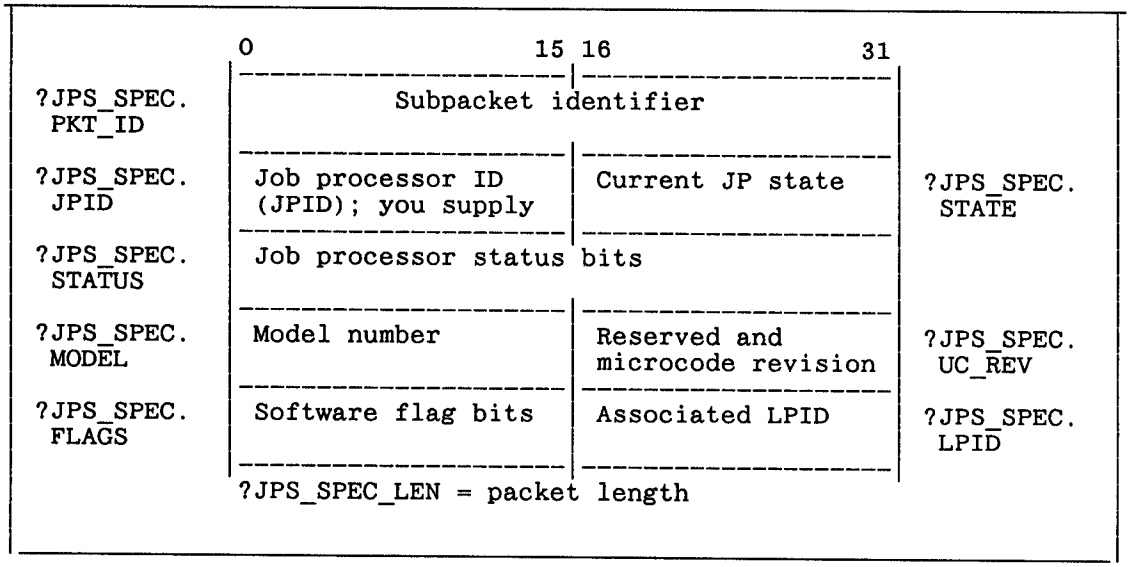


Figure 2-82. Structure of ?JPSTAT Specific Information Subpacket

Table 2-63. Contents of ?JPSTAT Specific Information Subpacket

Offset	Contents
?JPS_SPEC.PKT_ID (double word)	Subpacket identifier. Place ?JPS_SPEC_PKTID here.
?JPS_SPEC.JPID	ID number of the job processor that you want to receive information about.
?JPS_SPEC.STATE	Current state of the JP.
?JPS_SPEC.STATUS (double word)	JP status bits.
?JPS_SPEC.MODEL	JP model number.
?JPS_SPEC.UC_REV	Left byte is reserved; right byte contains the microcode revision number.
?JPS_SPEC.FLAGS	Software flag bits. If bit ?JPS_SPEC.FLAGS_MOM is set, then the JP is the mother processor. If bit ?JPS_SPEC.FLAGS_SYSTSK is set, then the JP is running a system task.
?JPS_SPEC.LPID	ID number of the logical processor that is associated with the job processor you specified in offset ?JPS_SPEC.JPID.

?KCALL Keeps the calling resource and acquires a new resource. (16–Bit Processes Only)

?KCALL *[procedure entry]*
normal return
alternate return

Input

The OS passes the input accumulators and the carry bit to the new procedure

Output

The accumulators and the carry bit are unchanged, unless they are modified by the new procedure

Error Codes in AC0

The following error codes can be passed to the ?BOMB routine:

ERICM	Illegal system command
ERLRF	Overlay load error
ERSEN	Invalid shared library reference (The OS does not support shared libraries.)

Why Use It?

In general, you issue ?KCALL instead of ?RCALL only when you want to avoid releasing the calling resource; for example, to guarantee the correct return address within a movable resource.

What It Does

?KCALL keeps the calling resource, loads a new resource, and transfers control to the specified procedure entry in the new resource. You can pass the procedure entry as an argument to ?KCALL or pass it on the stack (via the .PTARG pseudo–op). If you pass the procedure entry on the stack, the operating system pops it off as it executes ?KCALL.

After the operating system executes the target procedure, it returns control to the normal return, unless the target procedure specified an alternate return by issuing an ISZ ?ORTN,3 instruction followed by an RTN instruction.

If the calling task terminates, the operating system releases the newly acquired resource, but does not release the calling task. Therefore, do not terminate a task until all its outstanding ?KCALLs are completed.

Because ?KCALL does not release the calling resource, it requires more memory than ?RCALL. ?KCALL can also cause resource deadlocks. We recommend that you use ?RCALL instead of ?KCALL in most cases.

Notes

- See the description of ?RCALL in this chapter.

?KHIST

Kills a histogram.

?KHIST
error return
normal return

Input

None

Output

None

Error Codes in AC0

ERHIS Error on histogram init/delete (caller has no active histogram)

Why Use It?

?KHIST explicitly stops histogram monitoring of a process. If you do not issue ?KHIST, the histogram continues until the process that initiated it terminates.

What It Does

?KHIST stops the histogram started by a previous ?WHIST or ?IHIST system call. ?KHIST requires no input parameter, because a process can run only one histogram at a time.

Notes

- See the description of ?IHIST and ?WHIST in this chapter.

?KILAD — Defines a kill–processing routine.

?KILAD
error return
normal return

Input

AC0	Address of the kill–processing routine
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

In general, you define a kill–processing routine to release a task's resources in an orderly fashion when it terminates. For example, a kill–processing routine might consist of stack release instructions and explicit instructions to release any channels or user devices that are currently open. You should end a kill–processing routine with ?KILL to terminate the calling task.

?KILAD defines a kill–processing routine for the calling task only. You can also set up a generalized kill–processing routine, ?UKIL, for all tasks in a process. You can execute both ?KILAD and ?UKIL processing for any task.

What It Does

?KILAD defines a kill–processing routine for the calling task. The kill–processing routine gains control the first time another task issues ?IDKIL or ?PRKIL to terminate the ?KILAD caller. (The operating system ignores the kill–processing routine when a task terminates with ?KILL or with a WRTN instruction.)

Before you issue ?KILAD, load AC0 with the address of the kill–processing routine (defined elsewhere in your program). When a task enters a kill–processing routine, the operating system assigns it the highest priority level (0) so that it will gain control as soon as possible. (It may gain control immediately, if there are no other tasks at priority level 0.)

Notes

- See the descriptions of ?IDKIL, ?PRKIL, and ?KILL in this chapter.

?KILL

Kills the calling task.

?KILL
no error return
no normal return

Input

None

Output

None

Error Codes in AC0

ERLTK Last task was killed

Why Use It?

?KILL terminates the task that is currently executing. ?KILL also moves the terminated task's system databases to the inactive queue for eventual use in initiating another task.

What It Does

?KILL invokes the ?UKIL task–termination routine (either the user–defined ?UKIL or the default), and terminates the calling task unconditionally. ?KILL invokes the ?UKIL task–termination routine only on behalf of tasks that initiated processing within Ring 7 (that is, tasks whose initial PCs were Ring 7 addresses). When you use ?KILL to kill the last active, ready task in the process, the operating system returns error code ERLTK to AC0.

You can use ?KILL to terminate only the calling task. (The ?IDKIL and ?PRKIL system calls allow you to terminate the calling task or any other task that you specify by TID and priority, respectively.)

Notes

- See the descriptions of ?KILAD, ?IDKIL, and ?PRKIL in this chapter.

?KINTR Simulates keyboard interrupt sequences.

?KINTR
error return
normal return

Input

AC0	Control characters
AC1	PID to interrupt (must be a customer)
AC2	Unused

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes

ERPRH	Process is not in hierarchy (The PID that you tried to interrupt is not a customer.)

What It Does

?KINTR simulates keyboard interrupt sequences between processes that share a customer/server connection when the customer is waiting because of a ?KWAIT system call.

Why Use It?

?KINTR performs remote console interrupts. This allows virtual consoles to behave as if they were real consoles.

Notes

- See the description of ?KWAIT in this chapter.

?KIOFF Disables control–character console interrupts.

?KIOFF
error return
normal return

Input

AC0	Unused
AC1	Unused
AC2	Unused

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes

None

Why Use It?

?KIOFF prevents a process from being interrupted by a console control sequence.

What It Does

?KIOFF allows you to disable all two–control–character, console–interrupt sequences.

Notes

- See the description of ?KION in this chapter.

Licensed Material – Property of Data General Corporation

?KION

Re–enables control–character console interrupts.

?KION
error return
normal return

Input

AC0	Unused
AC1	Unused
AC2	Unused

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes

No error codes are currently defined.

Why Use It?

?KION re–enables console interrupts that were previously disabled by the ?KIOFF system call.

What It Does

?KION allows you to re–enable all two–control–character, console–interrupt sequences.

Notes

- See the description of ?KIOFF in this chapter.

Licensed Material – Property of Data General Corporation

?KWAIT

Waits for a console interrupt.

?KWAIT
error return
normal return

Input

AC0	Unused
AC1	Unused
AC2	Unused

Output

AC0	Second character of a two-control-character, console-interrupt sequence
AC1	Unchanged
AC2	Unchanged

Error Codes

ERKAD	Interrupt task already defined

Why Use It?

?KWAIT allows a process to handle console interrupts in nontraditional ways. This means that the same console interrupt can be handled in two different ways: the normal way or, if you issue ?KWAIT, another way.

What It Does

?KWAIT allows a process to wait for a console interrupt. When the interrupt occurs, the operating system passes the second character of the console control sequence to the waiting process. The waiting process can then take whatever action the control character directed.

Notes

- See the description of ?KINTR in this chapter.

Licensed Material - Property of Data General Corporation

?LABEL Creates a label for a magnetic tape.

?LABEL *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?LABEL packet

Error Codes in AC0

EREO1	File is open, can't exclusively open
EREO2	File is exclusively opened, can't open
ERFAD	File access denied
ERMPR	System call parameter address error
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?LABEL allows you to label a magnetic tape volume. (Another way to do this is to use the CLI LABEL utility.)

What It Does

?LABEL allows you to label a magnetic tape under program control. To use ?LABEL, you must supply a packet ?LBLN words long. You can specify the packet address in AC2 before you issue ?LABEL or you can use the packet address as an argument. Figure 2-83 shows the structure and Table 2-64 describes the contents of the ?LABEL packet.

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.

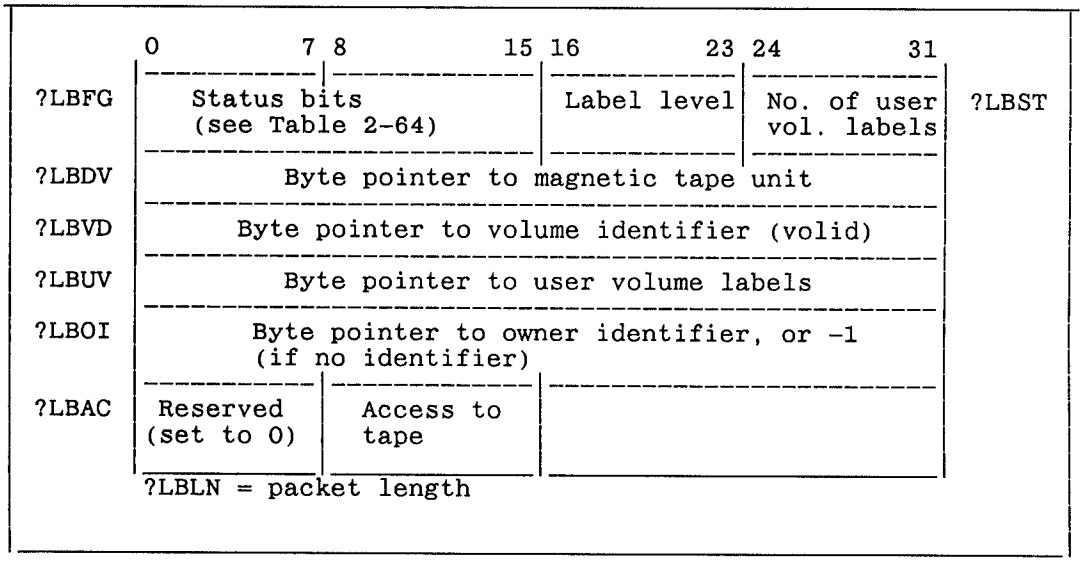


Figure 2-83. Structure of ?LABEL Packet

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 as it supplies the label.

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

Table 2-64. Contents of ?LABEL Packet*

Offset	Contents
?LBFG	Status bits ?LBIM--Label is in IBM format. ?LBSC--Scratch this tape. Density mode: ?LB8--800 bytes/inch. ?LB16--1600 bytes/inch. ?LB62--6250 bytes/inch. ?LBAM--automatic density matching.
?LBST	Left byte: label level. Right byte: number of user volume labels.
?LBDV (double word)	Byte pointer to magnetic tape unit.
?LBVD (double word)	Byte pointer to volume identifier (volid).
?LBUV (double word)	Byte pointer to user volume labels.
?LBOI (double word)	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.

Sample Packet

The following sample packet labels a tape so that everyone can use it:

```
PKT:    .BLK    ?LBLN           ;Allocate enough space for packet.
                                ;Packet length = ?LBLN.
        .LOC    PKT+?LBFG       ;Status bits and/or density mode.
        .WORD   ?LBAM           ;Set correct density automatically.
        .LOC    PKT+?LBST       ;Left byte: label level; right byte:
                                ;number of user volume label.
        .WORD   0               ;Default.
        .LOC    PKT+?LBDV       ;Byte pointer to magnetic tape unit.
        .DWORD  MAG*2           ;Byte pointer to MAG.
        .LOC    PKT+?LBVD       ;Byte pointer to volid.
        .DWORD  VOL*2           ;Byte pointer to VOL.
        .LOC    PKT+?LBUV       ;Byte pointer to user volume labels.
        .DWORD  UVL*2           ;Byte pointer to UVL.
        .LOC    PKT+?LBOI       ;Byte pointer to owner identifier.
        .DWORD  -1              ;No owner identifier.
        .LOC    PKT+?LBAC       ;Access to tape.
        .WORD   <40>            ;Everyone can use it (ASCII 40 =
                                ;space).
        .LOC    PKT+?LBLN       ;End of packet.
```

Notes

- See the description of ?OPEN in this chapter.
- Refer to your operating system's "How to Generate and Run" manual for information on the system-generation procedure.

?LEFD — Disables LEF mode.

?LEFD
error return
normal return

Input

None

Output

AC0	Contents lost
AC1	Undefined
AC2	Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

By default, each process begins executing with the CPU in LEF mode. For a process to perform I/O, LEF mode must be disabled (and I/O mode enabled). Thus, you must issue ?LEFD before you issue primitive I/O instructions; that is, I/O instructions distinct from the I/O system calls.

What It Does

?LEFD disables LEF mode (load-effective address mode) for the calling process. When LEF mode is disabled, the load-effective address instructions execute as I/O instructions. If LEF mode is already disabled when you issue ?LEFD, ?LEFD takes the normal return, but has no effect.

Note that although ?LEFD disables LEF mode, it does not automatically re-enable I/O mode. To disable LEF mode and re-enable I/O mode, issue ?DEBL.

Notes

- See the descriptions of ?LEFE, ?LEFS, and ?DEBL in this chapter.

?LEFE

Enables LEF mode.

?LEFE
error return
normal return

Input

None

Output

AC0	Contents lost
AC1	Undefined
AC2	Undefined

Error Codes in AC0

No error codes are currently defined.

What It Does

?LEFE enables LEF mode (load-effective address mode) for the calling process. LEF mode allows the caller to successfully issue LEF instructions. If LEF mode is already enabled when you issue ?LEFE, ?LEFE takes the normal return, but has no effect.

Why Use It?

?LEFE is the inverse of ?LEFD. You must issue ?LEFE if you want to re-enable LEF mode after you disable it with ?LEFD.

Notes

- See the descriptions of ?LEFD, ?LEFS, and ?DDIS in this chapter.

?LEFS Returns the current LEF mode status.

?LEFS
error return
normal return

Input

None

Output

AC0 One of the following:

- -1 if LEF mode is enabled
- 0 if LEF mode is disabled

AC1 Undefined

AC2 Undefined

Error Codes in AC0

No error codes are currently defined.

What It Does

?LEFS returns the current LEF-mode (load-effective address mode) status to AC0. If the CPU is in LEF mode, AC0 contains -1 on output. If the CPU is not in LEF mode, AC0 contains 0 on output.

No device I/O can occur while the CPU is in LEF mode, because LEF-mode instructions and I/O instructions use the same bit patterns. Similarly, when LEF mode is disabled, the operating system executes LEF instructions as if they are I/O instructions. Therefore, the state of the CPU determines whether the operating system executes LEF-mode or I/O-mode instructions.

Why Use It?

You can use ?LEFS to check the status of LEF mode.

Notes

- See the descriptions of ?LEFD and ?LEFE in this chapter.

?LMAP

Maps a lower ring.

?LMAP
error return
normal return

Input

AC0	Source address (page to map)
AC1	Page destination address (page to map)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERSNR	Source not resident
ERDNR	Destination not resident
ERPRV	Caller not privileged
ERMPR	Not an unshared page
ERPTY	Illegal process type

Why Use It?

?LMAP allows privileged programs to monitor system counters and other system status information.

What It Does

?LMAP allows a privileged user to remap portions of the process's address space.

To use ?LMAP, Superprocess mode must be turned on. The issuer of ?LMAP must be resident at the time the call is issued and must remain resident while the ?LMAP is in effect.

The source page can be in any ring of your address space, but it must be in memory (resident) at the time the ?LMAP call is issued, and, if the source address is not in Ring 0, the page must be wired. The source page cannot change its status (residency, wired/nonwired, current usage, etc.) in any way while it remains the source for the ?LMAP. It is up to the ?LMAP caller to ensure that its status does not change, as no record is made that the ?LMAP was performed.

The destination page for the ?LMAP must be an unshared, wired page when ?LMAP is issued. The system call removes the destination page from the working set of the process and replaces it with a read-only mapping of the source page. The ?LMAP destination page must not be referred to by any subsequent system call. It is up to the ?LMAP caller to ensure this, as there is no information in the system that the destination address is an ?LMAP page.

Warning

Failure to heed these restrictions may cause the operating system to PANIC or HANG. These potential problems restrict ?LMAP to Superprocess privilege holders.

?LOCALITY

Changes user locality.

AOS/VS only

?LOCALITY *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?LOCALITY packet

Error Codes in AC0

ERILV	Illegal locality value
ERPRH	Process not in hierarchy
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to change the user locality and class of a process.

What It Does

This system call allows a process to change the user locality (and thus class) of itself or of another process. When you change a process's user locality, AOS/VS indexes into the class matrix and calculates the new class ID for the process. AOS/VS then schedules the process on the basis of its new class.

You modify user locality for the calling process as follows. Place the value of ?LOC_PID in offset ?LOC_PKT.PCODE of the packet, and place -1 in offset ?LOC_PKT.PID.

You modify user locality for a different process in one of the two following ways:

- Assume that you know the name of the process. Place the value of ?LOC_PNAME in offset ?LOC_PKT.PCODE. Place in offset ?LOC_PKT.PNAME a byte pointer to the process name. Set offset LOC_PKT.PID to zero.
- Assume that you know the PID of the process. Place the value of ?LOC_PID in offset ?LOC_PKT.PCODE. Place zero in offset ?LOC_PKT.PNAME. Place in offset ?LOC_PKT.PID the PID.

You must have the following privileges in order to issue ?LOCALITY:

- If you have the system manager privilege ON, you may change the locality of any process on the system to any legal user locality value.
- If you do not have the system manager privilege ON, then
  - You may only issue ?LOCALITY against yourself or processes in your subtree (i.e., sons, grandsons, etc.).
  - Also, you may only change processes to those user localities designated at ?PROC time as legal for your process.

Figure 2-84 shows the structure of the ?LOCALITY parameter packet, and Table 2-65 describes its contents.

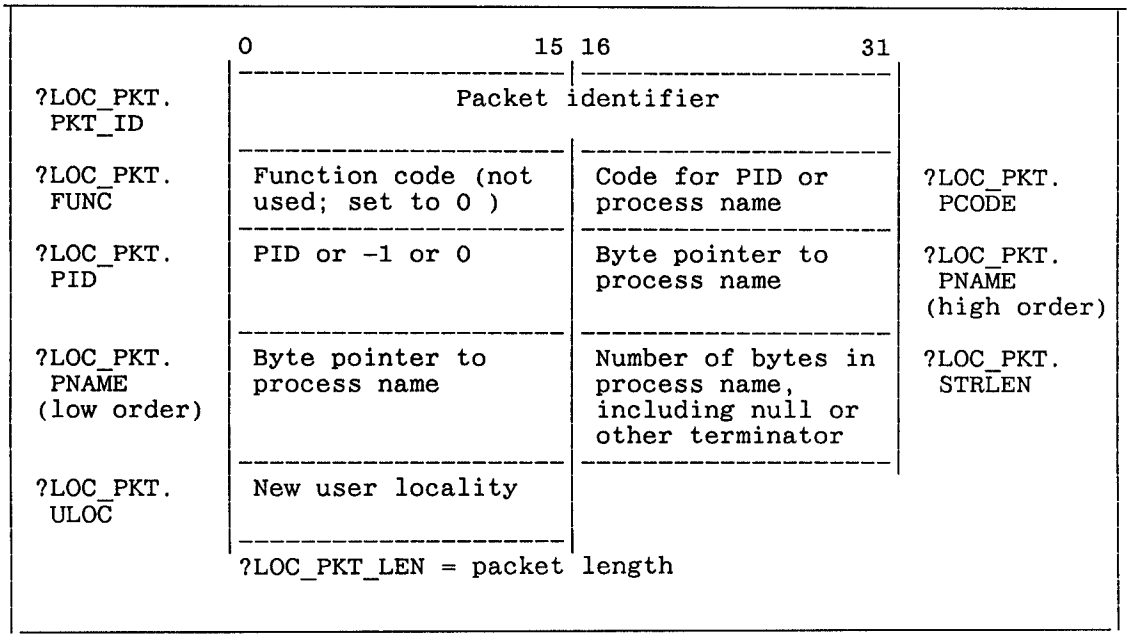


Figure 2-84. Structure of ?LOCALITY Packet

Table 2-65. Contents of ?LOCALITY Packet

Offset	Contents
?LOC_PKT.PKT_ID (double word)	Packet identifier. Place ?LOC_PKT_PKTID here.
?LOC_PKT.FUNC	Function code. Not used. (Set to 0.)
?LOC_PKT.PCODE	Code word into which you place ?LOC_PID when you supply a PID or else ?LOC_PNAME when you supply a process name.
?LOC_PKT.PID	Supply -1 for the calling process. For another process, place its PID here or else 0 here and the process's name information in the next two offsets.
?LOC_PKT.PNAME (double word)	If you've placed 0 in the previous offset, then place a byte pointer here to the process's name and place its length in the next offset.
?LOC_PKT.STRLEN	If you've placed 0 in offset ?LOC_PKT.PID and a byte pointer in the previous offset, then place the number of bytes in the byte pointer here.
?LOC_PKT.ULOC	Supply the new user locality.

?LOGCALLS — Logs system calls.

?LOGCALLS
error return
normal return

Input

AC0	Contains -1
AC1	Flag word: • ?LSTART — start system call logging • ?LFOP — start forced output
AC2	Byte pointer to pathname of file to create (ignored if ?LSTART in AC1 = 0)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERVBP Invalid byte pointer passed as a system call argument

File system error codes

Why Use It?

?LOGCALLS allows you to keep track of every system call that a particular process issues, including invalid system calls and system calls that inner rings issue.

You can use system call logging both as a debugging tool and to help you analyze a program's performance.

What It Does

?LOGCALLS logs system calls (including invalid system calls) into a file whose name you specify in AC2. To start logging, issue ?LOGCALLS with the ?LSTART flag set in AC1.

?LOGCALLS Continued

When you issue ?LOGCALLS to start logging after logging has already begun, the operating system closes the old log file and creates a new one. However, if you specify a file that already exists as a log file, ?LOGCALLS tries to delete the old file and create a new one with the same name. This allows you to periodically restart system call logging so that your log file contains only those system calls that you issued recently.

You can stop system call logging voluntarily by issuing ?LOGCALLS with the ?LSTART bit set to off in AC1. When a program terminates (via ?TERM to your process, ?RETURN, ?BRKFL, CTRL-C CTRL-B, or if a superior process terminates your process), system call logging stops involuntarily. No matter how system call logging stops, you can still read the log file.

If you try to stop system call logging, but it has already stopped or was never started, the operating system ignores all but the first ?LOGCALLS stop and returns normally.

You can minimize system write activity by setting the ?LFOP bit to 0 in AC1. This causes the operating system to buffer all output before it writes it to the log file. However, if the process is terminated without the ?LFOP bit set in AC1, the operating system may not write the last buffer of system calls to the log file. Therefore, if you want to make sure that the log file contains each system call when it is made, you can force output to the file without buffering by setting the ?LFOP bit in AC1.

To read the log file, you can use the LOGCALLS utility program, which prints the output in report form, or you can write your own program.

The system call log file consists of the following elements:

- Header record.

  The operating system creates the header record when system call logging starts.

- Detailed records for each system call.

  The record format is dynamic (?RTDY). Therefore, you must specify the length of the record that you want to read. To specify binary read, set the ?IBIN flag in the ?ISTI offset of the ?OPEN/?READ packet.

Header record entries in the system call log file contain ?LOGHREC words in the following format (offsets are in octal):

Words 0 through 5	Time of day that logging began
Words 0 through 1	Seconds
Words 2 through 3	Minutes
Words 4 through 5	Hours
Words 6 through 13	Day that logging began
Words 6 through 7	Day of month
Words 10 through 11	Month of year
Words 12 through 13	Year
Words 14 through 23	Process name in the following format: username:simple_processname The process name contains up to 15 characters and a null (ASCII 0) terminating character.
Word 24	Flag word Bit ?LOGF16U set means 16-bit process. Other bits are undefined.
Words 25 through 224	Pathname of .PR file that was loaded when you started system call logging. This pathname has up to 255 characters and a null (ASCII 0) terminating character.

The detailed record entries contain ?LOGDREC words in the following format (offsets are in octal):

Words 0 through 1	Contents of caller's AC0
Words 2 through 3	Contents of caller's AC1
Words 4 through 5	Contents of caller's AC2
Words 6 through 7	Location of the system call word (PC)
Words 10 through 11	Contents of caller's frame pointer
Word 12	Task identifier (TID)
Word 13	Unique TID
Word 14	System call number

Notes

- See the descriptions of ?RETURN, ?BRKFL, ?OPEN, and ?READ in this chapter.

?LOGEV Enters an event in the system log file.

?LOGEV
error return
normal return

Input

AC0	Event code you want to enter in the log file
AC1	Byte length of the message you want to enter in the log file
AC2	Byte pointer to the message you want to enter in the log file (ignored, if AC1 contains 0)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPRP	Illegal priority
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?LOGEV with ?SYLOG, which creates, renames, or checks the status of the system log file. Note that access to both ?LOGEV and ?SYLOG is restricted. Only a process with the Superuser privilege can issue ?LOGEV; only the operator process (PID 2) can issue ?SYLOG.

The system log file contains current information about all user console or batch processes, such as usernames, the times each user logged on and off, the devices each user accessed, and information about CPU and memory usage.

What It Does

?LOGEV allows a process with the Superuser privilege to enter an event code and/or a message into the system's log file, :SYSLOG. The operating system enters the log file data in the format shown in Figure 2-85.

The operating system records the event code, the length of the entire log entry, and the date and time it was entered in a packet of eight 16-bit words. However, the message text, if you choose to use it, can be any length. If necessary, the operating system pads the message text with zeros to a multiple of eight 16-bit words.

You can use the event-code field to assign each event a unique code, from ?LUMI through ?LMAX. Event codes ?LSMI and ?LDMA are reserved for system use.

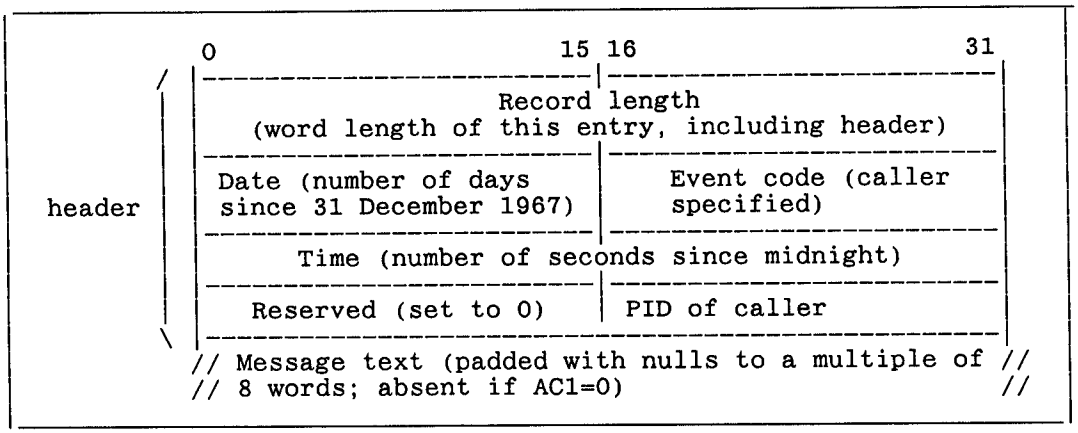


Figure 2-85. ?LOGEV Event Logging Format

The following restrictions apply to ?LOGEV:

- If AC1 (message length) = 0, the operating system ignores whatever message you specify and writes only the packet to the log file.
- The operating system writes only the first 760 (octal) bytes of the message to the log file. (If the message exceeds 760 (octal) bytes, the operating system ignores the remainder.)
- If the log function is turned off, the operating system ignores the message, but takes the ?LOGEV normal return.

Notes

- See the description of ?SYLOG in this chapter.
- See PARU.32.SR for a complete list of the event codes.
- Refer to your operating system's "How to Generate and Run" manual for information on accessing the system log file.

?LPCLASS

Gets/sets logical processor class assignments.

AOS/VS only

?LPCLASS *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?LPCLASS packet

Error Codes in AC0

ERCPC	Invalid class percentage value
ERHLP	Illegal hierarchy level/percentage pair
ERHLT	Nonsequential hierarchy levels designated
ERICD	Illegal function code
ERICI	Invalid class ID
ERITI	Invalid time interval
ERLNE	Logical processor (LP) does not exist
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to receive the current class scheduling data for a particular LP or to declare class scheduling information for a particular LP.

What It Does

To indicate whether you want to get current information about classes for an LP or want to set this information, specify a get/set code in the parameter packet.

The time interval word in the parameter packet specifies the amount of time over which the percentages (in the next words) will be spread. That is, AOS/VS attempts to schedule the various percentages over the specified time interval. The value is between ?LPCL_PKT.TIME_MIN and ?LPCL_PKT.TIME_MAX tenths of a second. At the time of this writing, the respective values were 1 and 100 (0.1 and 10.0 seconds). If you place ?LPCL_PKT.TIME_DEF in this word, you specify the default value of 40*0.1 seconds = 4.0 seconds.

Licensed Material - Property of Data General Corporation

The last portion of the packet is a set of 16 words that forms a table for scheduling each of the 16 possible classes. Each word contains a hierarchy level in its left byte and a percentage value in its right byte. Supplying a percentage value makes the corresponding class primary; AOS/VS attempts to give that class the specified percentage (between 1 and 100) of CPU time. Supplying a hierarchy level makes the corresponding class secondary; the supplied number indicates the hierarchy level of the class. AOS/VS only schedules secondary classes when there are no eligible primary classes ready to run. When this occurs, AOS/VS schedules the secondary classes in order of their hierarchy level (classes at hierarchy level 1 first, classes at hierarchy level 16 last). You cannot supply both a percentage value and a hierarchy level for the same class. That is, a class cannot be both primary and secondary within a single LP.

Be sure that your hierarchy levels are sequential starting with level 1 and increasing in a step of 1. For example, you may designate classes at hierarchy levels 1, 2, 3, and 4 — but not at hierarchy levels 1, 2, 3, and 5.

Figure 2-86 shows the structure of the ?LPCLASS parameter packet, and Table 2-66 describes its contents.

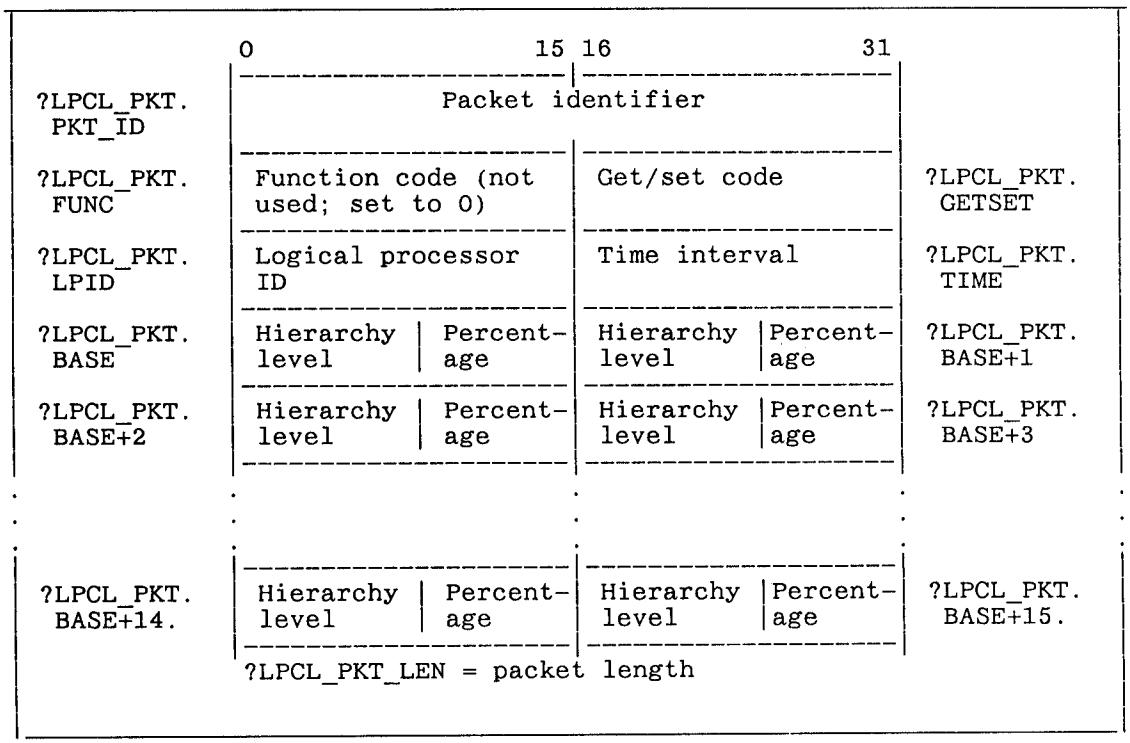


Figure 2-86. Structure of ?LPCLASS Packet

?LPCLASS Continued

Table 2-66. Contents of ?LPCLASS Packet

Offset	Contents
?LPCL_PKT.PKT_ID (double word)	Packet identifier. Place ?LPCL_PKT_PKTID here.
?LPCL_PKT.FUNC	Function code. Not used. (Set to 0.)
?LPCL_PKT.GETSET	Get/set code. You supply ?LPCL_GET to get class values or ?LPCL_SET to set class values.
?LPCL_PKT.LPID	Logical processor ID for the LP about which you will get class data or set class data.
?LPCL_PKT.TIME	Time interval, with the number of tenths of a second, during which you will get or set the schedule of classes.
?LPCL_PKT.BASE	Hierarchy level and percentage of CPU time (left byte/right byte) for class number 0.
?LPCL_PKT.BASE+1	Hierarchy level and percentage of CPU time (left byte/right byte) for class number 1.
?LPCL_PKT.BASE+2	Hierarchy level and percentage of CPU time (left byte/right byte) for class number 2.
?LPCL_PKT.BASE+3	Hierarchy level and percentage of CPU time (left byte/right byte) for class number 3.
.	.
.	.
.	.
?LPCL_PKT.BASE+14.	Hierarchy level and percentage of CPU time (left byte/right byte) for class number 14.
?LPCL_PKT.BASE+15.	Hierarchy level and percentage of CPU time (left byte/right byte) for class number 15.

?LPCREA — Creates a logical processor.

AOS/VS only

?LPCREA *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?LPCREA packet

Error Codes in AC0

ERMLP	Attempt to exceed maximum logical processor (LP) count
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to create a new logical processor (LP).

What It Does

This system call creates a new LP. AOS/VS creates an LP and returns its ID number (LPID). Then, you can set up LP class designations, assign job processors, etc.

AOS/VS assigns a value to LPID between ?LPID_MIN and ?LPID_MAX, respectively.

Licensed Material - Property of Data General Corporation

?LPCREA Continued

Figure 2-87 shows the structure of the ?LPCREA parameter packet, and Table 2-67 describes its contents.

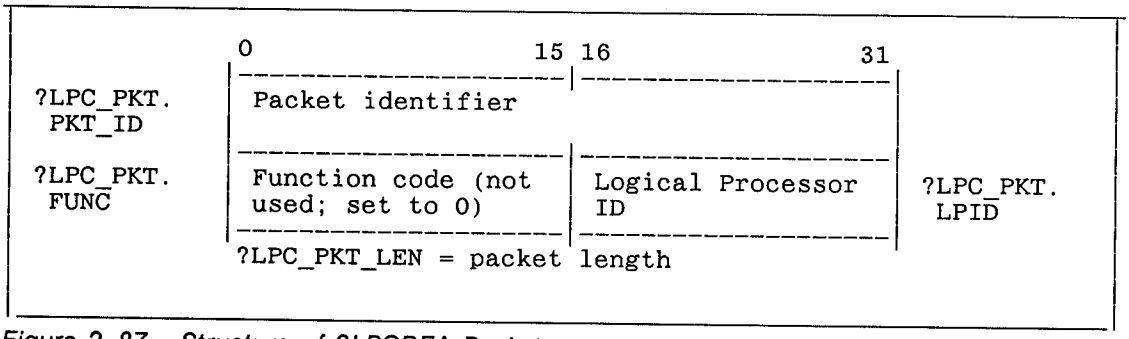


Figure 2-87. Structure of ?LPCREA Packet

Table 2-67. Contents of ?LPCREA Packet

Offset	Contents
?LPC_PKT.PKT_ID (double word)	Packet identifier. Place ?LPC_PKT_PKTID here.
?LPC_PKT.FUNC	Function code. Not used. (Set to 0.)
?LPC_PKT.LPID	Logical processor ID. AOS/VS returns this value.

?LPDELE — Deletes a logical processor.

AOS/VS only

?LPDELE *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?LPDELE packet

Error Codes in AC0

ERILP	Invalid LPID
ERJAA	Job processors (JP) already attached to logical processor (LP)
ERLNE	LP does not exist
ERLP0	Cannot delete LP 0
ERPRV	Caller not privileged for this action

Why Use It?

Use this system call to remove a logical processor (LP).

What It Does

This system call deletes an LP. However, you cannot delete an LP if there are any job processors (JPs) attached to it. You also cannot delete LP 0 — the LP that exists when your AOS/VS system first comes up.

?LPDELE Continued

Figure 2-88 shows the structure of the ?LPDELE parameter packet, and Table 2-68 describes its contents.

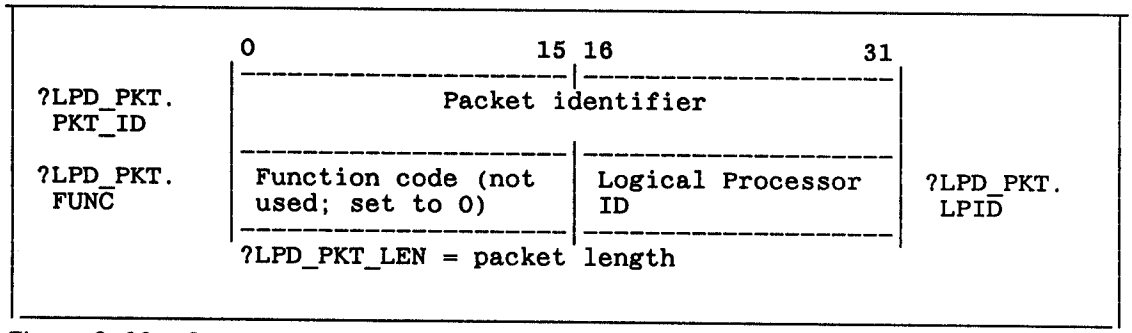


Figure 2-88. Structure of ?LPDELE Packet

Table 2-68. Contents of ?LPDELE Packet

Offset	Contents
?LPD_PKT.PKT_ID (double word)	Packet identifier. Place ?LPD_PKT_PKTID here.
?LPD_PKT.FUNC	Function code. Not used. (Set to 0.)
?LPD_PKT.LPID	Logical processor ID. You supply this value.

?LPSTAT Gets the status of a logical processor.

AOS/VS only

?LPSTAT *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?LPSTAT packet

Error Codes in AC0

ERICD	Invalid function code
ERILP	Invalid LPID
ERLNE	Logical processor (LP) does not exist

Why Use It?

Use this system call to obtain information about all current logical processors (LP) or about a specific current logical processor.

What It Does

This system call has two versions — general and specific. The general version returns the total number and LPIDs of all current logical processors. The specific version returns status information of a specific logical processor. You tell AOS/VS which version you want by placing a code in offset ?LPS_PKT.FUNC of the main parameter packet.

Licensed Material - Property of Data General Corporation

?LPSTAT Continued

If you select the general version, then AOS/VS returns, in a double word of the general status subpacket, a bit map that indicates what logical processors exist:

- If a bit is set, then the corresponding LP exists.
- If a bit is not set, then the corresponding LP does not exist.

For example, suppose bits 2, 3, and 5 are set. This means LPs with IDs 2, 3, and 5 exist. Furthermore, LPs with IDs 0, 1, 4, 6, 7, 8, ..., 15 do not exist.

If you select the specific version then AOS/VS returns, in a double word of the specific status subpacket, a bit map that indicates what job processors are associated with the given logical processor:

- If a bit is set, then the corresponding JP is associated with the LP.
- If a bit is not set, then no JP is associated with the LP.

For example, suppose bits 2, 3, and 5 are set. This means job processors (JPs) with IDs 2, 3, and 5 are associated with the specified LP. Furthermore, JPs with IDs 0, 1, 4, 6, 7, 8, ..., 15 are not associated with the specified LP.

Figure 2-89 shows the structure of the ?LPSTAT main parameter packet, and Table 2-69 describes its contents. Figure 2-90 and Figure 2-91 show the structure of the general status subpacket and specific status subpacket, respectively. Table 2-70 and Table 2-71 describe the contents of these respective subpackets.

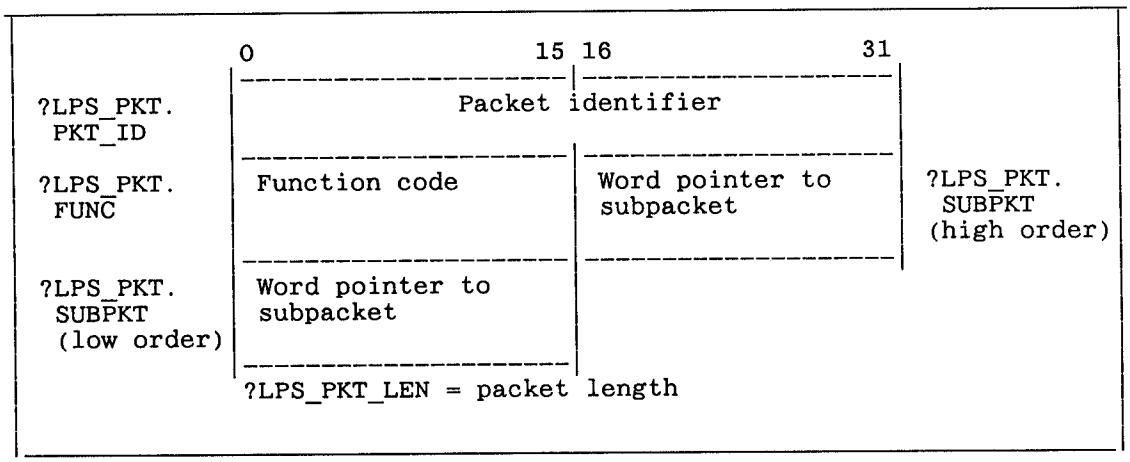


Figure 2-89. Structure of ?LPSTAT Main Packet

Table 2-69. Contents of ?LPSTAT Main Packet

Offset	Contents
?LPS_PKT.PKT_ID (double word)	Packet identifier. Place ?LPS_PKT_PKTID here.
?LPS_PKT.FUNC	General/Specific code. Place ?LPS_GEN here to obtain general information or ?LPS_SPEC to obtain specific information. Both values must be between ?LPS_FUNC_MIN and ?LPS_FUNC_MAX, inclusive.
?LPS_PKT.SUBPKT (double word)	If you have selected general information, place the word address of the general status subpacket here. If you have selected specific information, place the word address of the specific status subpacket here.

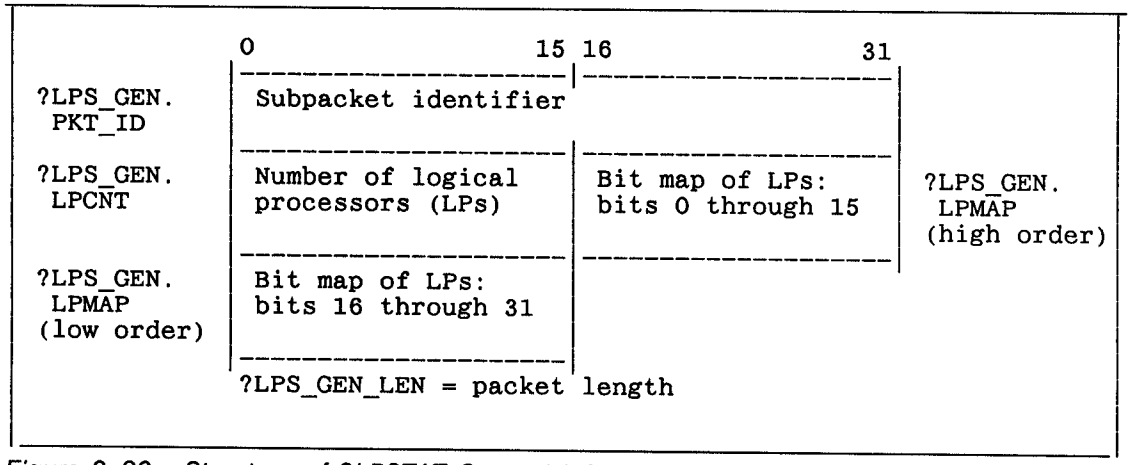


Figure 2-90. Structure of ?LPSTAT General Information Subpacket

Table 2-70. Contents of ?LPSTAT General Information Subpacket

Offset	Contents
?LPS_GEN.PKT_ID (double word)	Subpacket identifier. Place ?LPS_GEN_PKTID here.
?LPS_GEN.LPCNT	Number of logical processors that currently exist.
?LPS_GEN.LPMAP (double word)	Bit map to indicate the LPIDs of the logical processors that currently exist. For example, if AOS/VS has set the leftmost bit of the first (high-order) word in this double word, then logical processor number 0 currently exists.

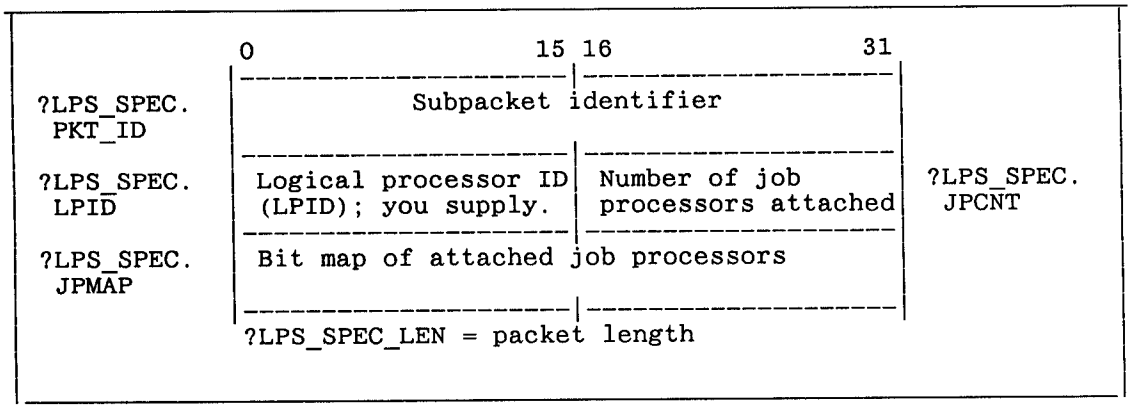


Figure 2-91. Structure of ?LPSTAT Specific Information Subpacket

Table 2-71. Contents of ?LPSTAT Specific Information Subpacket

Offset	Contents
?LPS_SPEC.PKT_ID (double word)	Subpacket identifier. Place ?LPS_SPEC_PKTID here.
?LPS_SPEC.LPID	ID number of the logical processor that you want to receive information about.
?LPS_SPEC.JPCNT	Number of job processors attached to the logical processor you specified in the previous word.
?LPS_SPEC.JPMAP (double word)	Bit map that indicates the JPIDs of the job processors attached to the logical processor you specified two words ago.

?MAPDV Maps a device into logical address space.

?MAPDV *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?MAPDV packet

Error Codes in AC0

ERPRE	Invalid system call parameter, for any of these reasons - Invalid ?MDPV packet revision (other than zero) - Invalid ?MDRT region type - Invalid ?MDRP starting page (negative) - Invalid ?MDPC number of pages to map
ERMPR	Invalid parameter address (bad packet pointer, etc.)
ERMSI	Map specification illegal for target (not a memory mapped device)
ERNMT	Map target does not exist
ERRAU	Region already in use

Why Use It?

Use this system call to map a region of a process's address space to a special memory mapped device.

What It Does

A memory mapped device (?MMAP) is any I/O device that your process can gain access to by reading or writing to a physical page that resolves to the device. A DS/7000-series I/O controller connected to the CPU chassis has a device code associated with it and a node number (determined by the slot the board resides in). DS/7000-series computers include DS/7500 series computers and the MV/2000 DC computer.

You supply to ?MAPDV a device code, number of pages to map, and logical address to map. ?MAPDV returns the physical address of the first page that it has assigned to that device code. This physical address includes the node associated with the device. You need the node number to gain access to special space to control the device.

?MAPDV performs the functionality of ?STMAP. ?STMAP deals with data channel mapped devices; ?MAPDV deals with memory mapped devices.

Controlling a Memory Mapped I/O device

The following procedure controls a memory mapped I/O device.

1. Issue ?IDEF and specify a programmable I/O (PIO) device, with no map definition table.
2. Issue ?MAPDV with the following packet offsets:
   - ?MDPK/?MDPL = ?MAPDV_PKT_PKTID.
   - ?MDRT = ?MMAP.
   - ?MDOP = ?MRDO.
   - ?MDIL = 6-bit device code.
   - ?MDRP/?MDRL = 32-bit starting page number (0 = first physical page in the I/O controller).
   - ?MDPC/?MDCL = 32-bit number of pages you need to map.
   - ?MDLA/?MDDL = 32-bit logical address.

     Returned offset ?MDN0/?MDN1 contains the node number needed to gain access to the special space to control the device.

     Returned offset ?MDDT/?MDDL contains the device type (model number) as returned from the SCP command SIZE MEMORY.
3. Do I/O processing as desired.
4. Issue ?MAPDV again with ?UNMR in offset ?MDOP. This is the only offset you have to change.
5. Issue ?IRMV to remove the device specified in offset ?MDDT/?MDDL.

Suppose a process terminates before ?MAPDV, which has ?UNMR in offset ?MDOP, is finished. ?IRMV will ensure that this mapped region is undone. If a process terminates before ?IRMV is finished, then the system will clean up after the termination.

?MAPDV Continued

Figure 2–92 shows the structure of ?MAPDV's parameter packet, and Table 2–72 describes its contents.

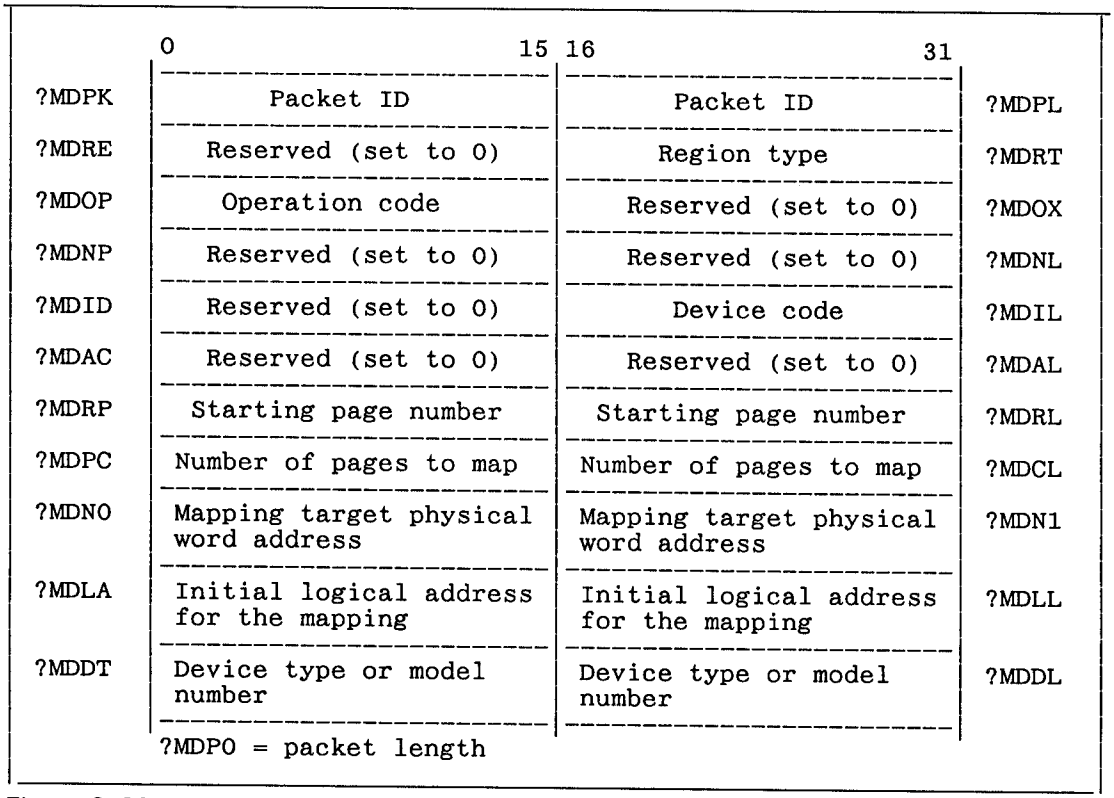


Figure 2–92. Structure of ?MAPDV Packet

Table 2-72. Contents of ?MAPDV Packet

Offset	Contents
?MDPK/ ?MDPL	Packet revision number. Place ?MAPDV_PKT_PKTID here.
?MDRE	Reserved. Set to 0.
?MDRT	Region type. The values and their meanings follow. ?BITM -- Physical video bit map memory. ?VBITM -- Virtual bit map memory. ?MMAP -- Memory mapped device memory.
?MDOP	Operation code. The values and their meanings follow. ?MRDO -- Add region to address space. ?UNMR -- Remove region from address space.
?MDOX	Reserved. Set to 0.
?MDNP/ ?MDNL	Reserved. Set to 0.
?MDID	Reserved. Set to 0.
?MDIL	Supply the target device's 6-bit code.
?MDAC/ ?MDAL	Reserved. Set to 0.
?MDRP/ ?MDRL	Supply the starting page number in the target. (0 is the first physical page in the I/O controller.)
?MDPC/ ?MDCL	Supply the number of pages ?MAPDV will map.
?MDN0/ ?MDN1	?MAPDV returns the physical address of the first page that is assigned to the device code of the I/O controller. This address includes the node associated with the target device.
?MDLA/ ?MDLL	Supply the initial logical address for the mapping.
?MDDT/ ?MDDL	?MAPDV returns the device type (model number).

?MBFC Moves bytes from a customer's buffer.

?MBFC *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Actual number of bytes moved if ?MBFC takes an error return
AC2	Address of the ?MBFC packet

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection doesn't exist
ERMPR	System call parameter address error
ERPRV	Caller not privileged for this action (The caller is not a server of the specified customer.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?MBFC and its companion system call, ?MBTC, give a server process access to a customer's logical address space. These system calls are useful for any application in which the server must move bytes to fulfill a customer's request.

What It Does

?MBFC moves a specific number of bytes from a buffer in a customer's logical address space to a buffer in the server's logical address space. The calling process must be a server of the designated customer.

You can use ?MBFC to move bytes from inner-ring customer buffers, but the address that you specify within the caller's process must be within a ring whose number is greater than or equal to the caller's ring. Also, all source data must reside entirely within the single specified source ring. There must be enough room at the destination for the data to reside entirely within the destination ring specified.

Figure 2-93 shows the structure of the move bytes packet, which both ?MBFC and ?MBTC use. The symbol ?MBLTH represents the length of the packet. Either load the packet address into AC2 before you issue ?MBFC, or cite its address as an argument to ?MBFC.

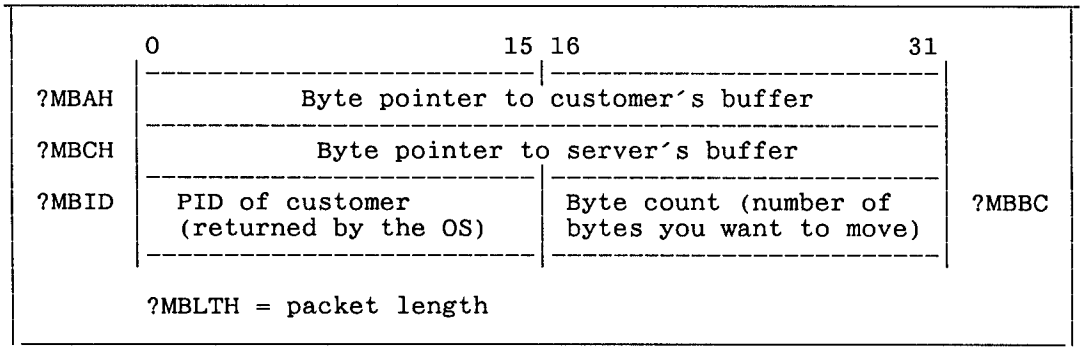


Figure 2-93. Structure of ?MBFC Packet

You can move a maximum of 2048 bytes with ?MBFC. If you specify a byte length longer than 2048, the operating system takes the error return on error code ERMPR, and returns 0 to AC1 (that is, moves 0 bytes). If the server's buffer is too small to accommodate all of the bytes, the operating system moves as many as it can and returns the number of bytes moved to AC1 and error code ERMPR to AC0.

Do not use ?MBFC against a customer that has chained to a new program, unless you can verify the validity of the customer's buffer. Similarly, customer processes should not issue ?CHAIN while they have outstanding requests to servers.

Sample Packet

The following sample packet is set up to move bytes from a customer's buffer:

```
PKT:    .BLK    ?MBLTH          ;Allocate enough space for the
                                ;packet.  Packet length = ?MBLTH.
        .LOC    PKT+?MBAH       ;Byte pointer to customer's buffer.
        .DWORD  CUSBUF*2        ;Customer's buffer is CUSBUF.
        .LOC    PKT+?MBCH       ;Byte pointer to server's buffer.
        .DWORD  SRVBUF*2        ;Server's buffer is SRVBUF.
        .LOC    PKT+?MBID       ;PID of customer.
        .WORD   14.             ;PID of customer is 14.
        .LOC    PKT+?MBBC       ;Number of bytes you want to move.
        .WORD   2048.           ;Move 2048 bytes.  (This is the
                                ;maximum you can transfer.)
        .LOC    PKT+?MBLTH      ;End of packet.
```

Notes

- See the descriptions of ?MBTC and ?CHAIN in this chapter.

?MBTC Moves bytes to a customer's buffer.

?MBTC *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Actual number of bytes moved if ?MBTC takes an error return
AC2	Address of the ?MBTC packet

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection does not exist
ERMPR	System call parameter address error
ERPRV	Caller not privileged for this action (The caller is not a server of the specified customer.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can use ?MBTC to move bytes from inner-ring customer buffers.

What It Does

?MBTC moves bytes from a buffer in the server's logical address space to a buffer in the designated customer's logical address space. The calling process must be a server of the target customer.

You can use ?MBFC to move bytes from inner-ring customer buffers, but the address that you specify within the caller's process must be within a ring whose number is greater than or equal to the caller's ring. Also, all source data must reside entirely within the single specified source ring. There must be enough room at the destination for the data to reside entirely within the destination ring specified.

?MBTC takes the same packet as ?MBFC. (See Figure 2-94.) You can cite the packet address as an argument to ?MBTC, or you can load its address into AC2 before you issue the system call.

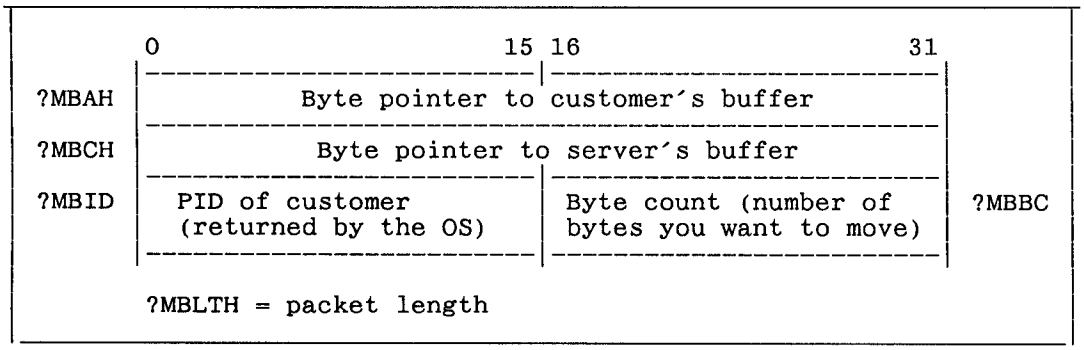


Figure 2-94. Structure of ?MBTC Packet

You can move a maximum of 2048 bytes with ?MBTC. If you specify a larger byte count, ?MBTC fails on error code ERMPR, and the operating system returns 0 to AC1 (0 bytes moved). If the customer's buffer is too small to accommodate the bytes, the operating system moves as many as it can; it returns the number of bytes moved to AC1 and error code ERMPR to AC0.

Do not use ?MBTC against a customer that has chained to a new program, unless you can guarantee the validity of the customer's buffer. (The same restriction applies to ?MBFC.)

Notes

- See the descriptions of ?MBFC and ?CHAIN in this chapter.

?MDUMP

Dumps the memory image from a user–specified ring to a file.

?MDUMP
error return
normal return

Input

AC0	Address that contains ring field of ring to be dumped
AC1	Reserved (Set to 0.)
AC2	One of the following: • Byte pointer to the dump file pathname • −1 to use the default pathname

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

File system error codes

Why Use It?

?MDUMP allows you to examine your program memory image when you are not logged on. ?MDUMP complements ?BRKFL, because the dump file contains all of the information in a break file, plus a memory image.

What It Does

?MDUMP allows you to obtain a snapshot dump of one ring of your address space while your program is executing. The operating system strips the ring field from AC0 and dumps that ring to the dump file that you specify.

The dump file is formatted much like a .PR file in that it has a preamble at the beginning and your actual memory image starts at location 20000. The preamble contains a dump of your task control blocks, which reside in Ring 3 as part of the agent. You can use the File Editor utility (FED) to examine the dump file. Also, you can run BRAN on the dump file.

Note that ?MDUMP checks to see whether you have Read access to the .PR file in the ring that you want to dump. If you do not have Read access, the operating system returns an error.

The default pathname of the dump file is in the form

?PID.TIME.RING.MDM

where

- PID is the 5-digit PID of the user.
- TIME is the current time in the form hours_minutes_seconds.
- RING is the ring number of the ring dumped.

Notes

- See the description of ?BRKFL in this chapter.

?MEM Lists the current unshared memory parameters.

?MEM
error return
normal return

Input

None

Output

AC0 Maximum number of unshared memory pages currently available

AC1 Number of unshared memory pages currently in use

AC2 Highest unshared address in the caller's logical address space (This value is always "page-adjusted", that is, the OS rounds this value by setting the 10 low-order bits to 1.)

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?MEM is a useful preliminary system call for ?MEMI. Depending on your input parameters, ?MEMI either increases or decreases the number of unshared pages in the logical address space.

Because ?MEM also returns the number of unshared memory pages currently available, you can also use ?MEM to determine your program's unshared size.

What It Does

?MEM returns the following unshared memory parameters:

- The maximum number of unshared pages available in the caller's logical address space.
- The number of unshared pages currently in use.
- The highest unshared address in the caller's logical address space.

Notes

- See the description of ?MEMI in this chapter.

?MEMI Changes the number of unshared pages in the logical address space.

?MEMI
error return
normal return

Input

AC0 One of the following:

- Number of unshared pages you wish to allocate
- Two's complement of the number of unshared pages you wish to remove

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Highest logical address in the caller's unshared logical address space

AC2 Undefined

Error Codes in AC0

ERMEM Insufficient memory available
ERMRL Memory release error (You tried to remove more pages than the unshared area contains, or you tried to remove Page 0.)

Why Use It?

?MEMI lets you expand the unshared area of the logical address space by using any unused space between the unshared and shared areas. You can also use ?MEMI to reduce the number of pages in the unshared area.

?MEMI Continued

What It Does

?MEMI increases or decreases the unshared area in the calling process's logical address space by the number of pages that you specify in AC0. The operating system returns the new maximum unshared address to AC1. Note that pages acquired with ?MEMI initially contain all zeros.

To increase the number of unshared pages, load AC0 with a positive value that indicates the number of additional pages you want. To decrease the number of unshared pages, load AC0 with the appropriate two's complement (negative) value.

You can't remove Page 0 of your logical address space.

?MEMI always takes or adds pages at the end of the unshared area. You can't have "holes" in your unshared address space.

Notes

- See the description of ?IREC in this chapter.

Licensed Material - Property of Data General Corporation

?MIRROR Mirrors and synchronizes LDU images.

AOS/VS only

?MIRROR *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?MIRROR packet

Error Codes in AC0

ERCNM	Controller does not support logical disk unit (LDU) mirroring
ERLDR	LDU released during synchronization
ERLDX	LDU does not exist
ERLFM	LDU format mismatch — not a valid LDU mirror
ERLMM	LDU name mismatch — not a valid LDU mirror
ERLSZ	LDU size mismatch — not a valid LDU mirror
ERMOP	Mirrored LDU is out of phase
ERRID	LDU IDs are not unique — not a valid LDU mirror
ERSFL	Mirrored LDU synchronization failed
ERTMM	Controller cannot support an additional LDU mirror

Licensed Material - Property of Data General Corporation

Why Use It?

You use ?MIRROR to manage your mirrored LDU (logical disk unit) configuration. The call controls all changes in the state of mirroring.

What It Does

?MIRROR allows you to add an image in a mirrored LDU configuration. LDU mirroring allows you to maintain two images of the same LDU. If one LDU image becomes unavailable, you still have access to the LDU data on the remaining available image.

Adding an image to a mirrored LDU requires that ?MIRROR synchronize the images so that they contain the same data. When ?MIRROR performs this synchronization, you direct it to either wait for the synchronization to complete or just invoke the synchronization.

Issuing ?MIRROR might use significant system resources when you synchronize a mirror. The related system task will use appreciable I/O resources and memory/CPU resources.

The caller's privileges must include Owner access to the target LD's root. The privileges must also include Execute access to the physical disks specified. ?MIRROR's privileges are similar to the ones that system call ?XINIT requires.

?MIRROR operates on LDU images. You frequently issue ?MIRROR along with ?XINIT and ?RELEASE since these two system calls also operate on LDU images.

Figure 2-95 shows the structure of the ?MIRROR main packet, and Table 2-73 describes its contents. Figure 2-96 shows the structure and contents of the ?MIRROR subpacket. This subpacket is identical to the one that ?XINIT uses except for the value of the packet identifier.

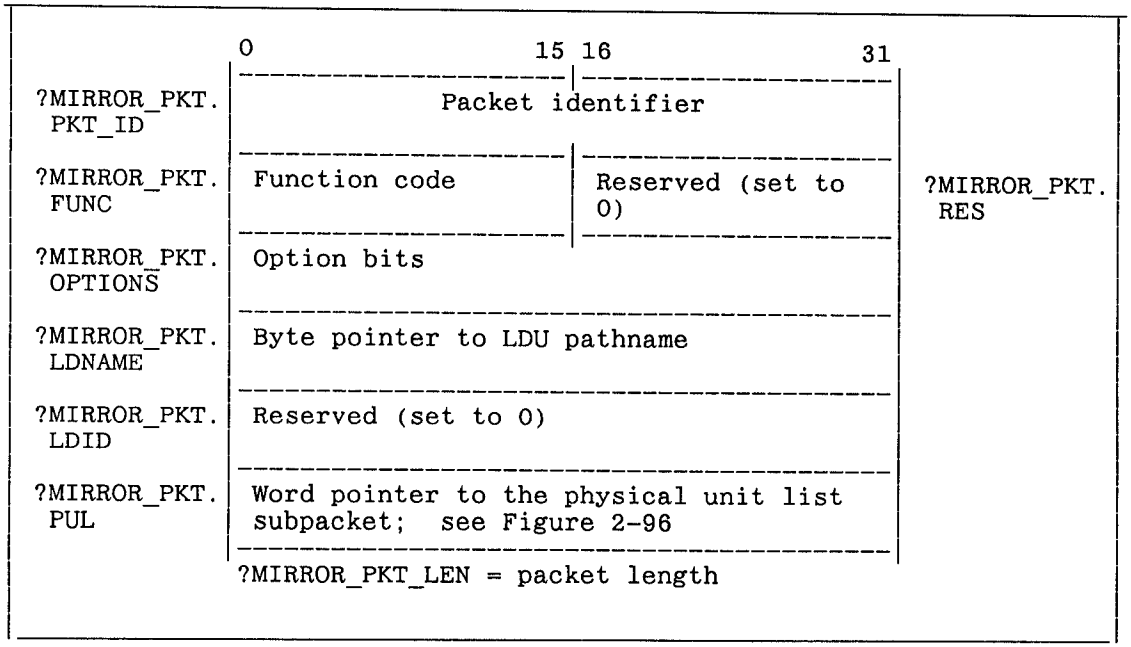


Figure 2-95. Structure of ?MIRROR Packet

Table 2-73. Contents of ?MIRROR Packet

Offset	Contents
?MIRROR_PKT.PKT_ID (double word)	Packet identifier. Place ?MIRROR_PKT_PKTID here.
?MIRROR_PKT.FUNC	Function code. Place the following value here: - ?MIRROR_SYNCHRONIZE to synchronize the mirror (i.e., add an image to the mirrored set.)
?MIRROR_PKT.RES	Reserved. (Set to 0.)
?MIRROR_PKT.OPTIONS (double word)	Options word. Choose from the following values here, but only when offset ?MIRROR_PKT.FUNC contains ?MIRROR_SYNCHRONIZE: - ?MIRROR_OVERRIDE to override the error condition ERMOP. ("Mirrored LDU is out of phase.") - ?MIRROR_SYNC_WAIT to wait for the synchronization process to complete. (The system might return error code ERSFL or ERLDR.)
?MIRROR_PKT.LDNAME (double word)	Supply a byte pointer to the pathname of the LDU name that the ?MIRROR call addresses. If specified LDU image is not associated with the specified LDU name, the system returns an error.
?MIRROR_PKT.LDID (double word)	Reserved. (Set to 0.)
?MIRROR_PKT.PUL (double word)	Supply a word pointer to the physical unit list subpacket. Figure 2-96 describes this subpacket.

?MIRROR Continued

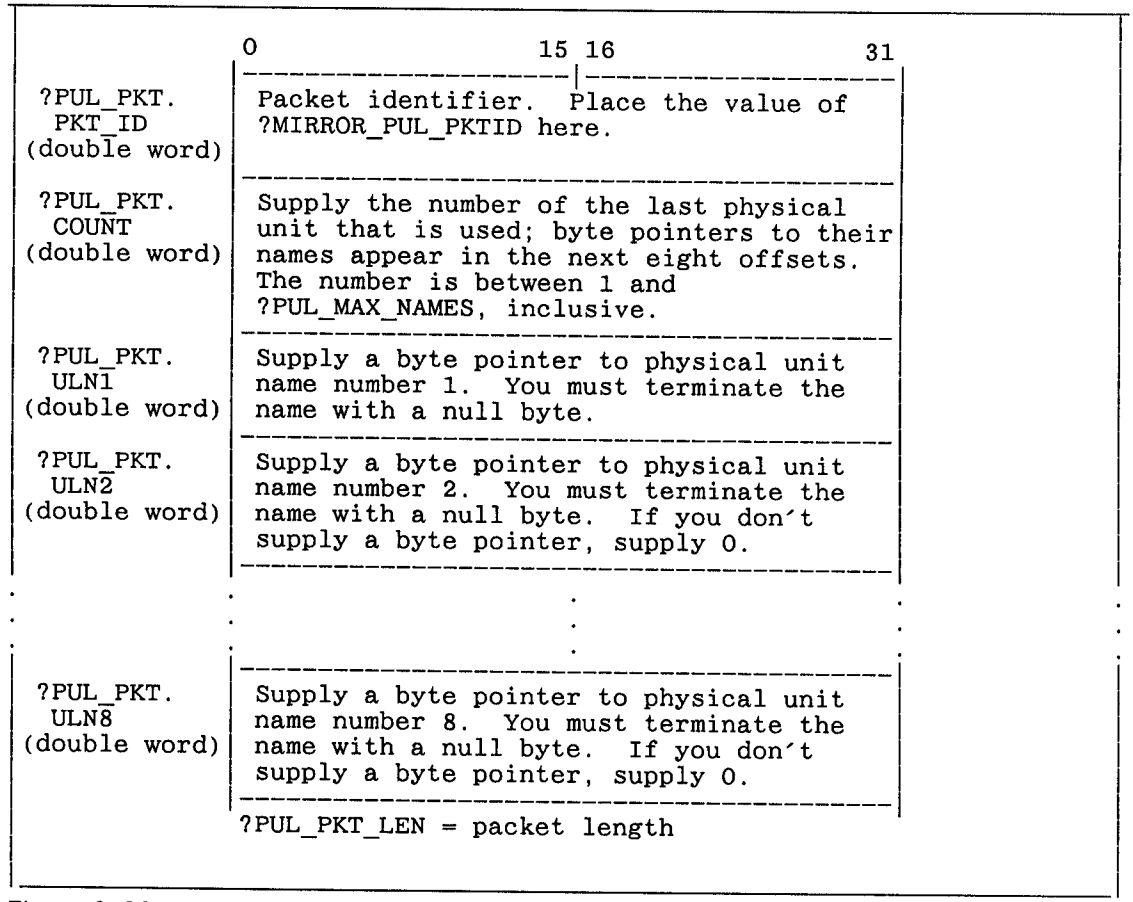


Figure 2-96. Structure of ?MIRROR Subpacket

16-bit Version of ?MIRROR System Call

A 16-bit version of this system call also exists. Its functionality is identical to that of the 32-bit version previously described.

Figure 2-97 contains the structure of the 16-bit ?MIRROR call main packet and Figure 2-98 contains the structure of the 16-bit ?MIRROR call subpacket. The definitions of the offsets in these packets are identical to those in Figure 2-95 and Figure 2-96, respectively. The only differences in the corresponding packets are the names of the offsets.

Table 2-74 describes the contents of Figure 2-97.

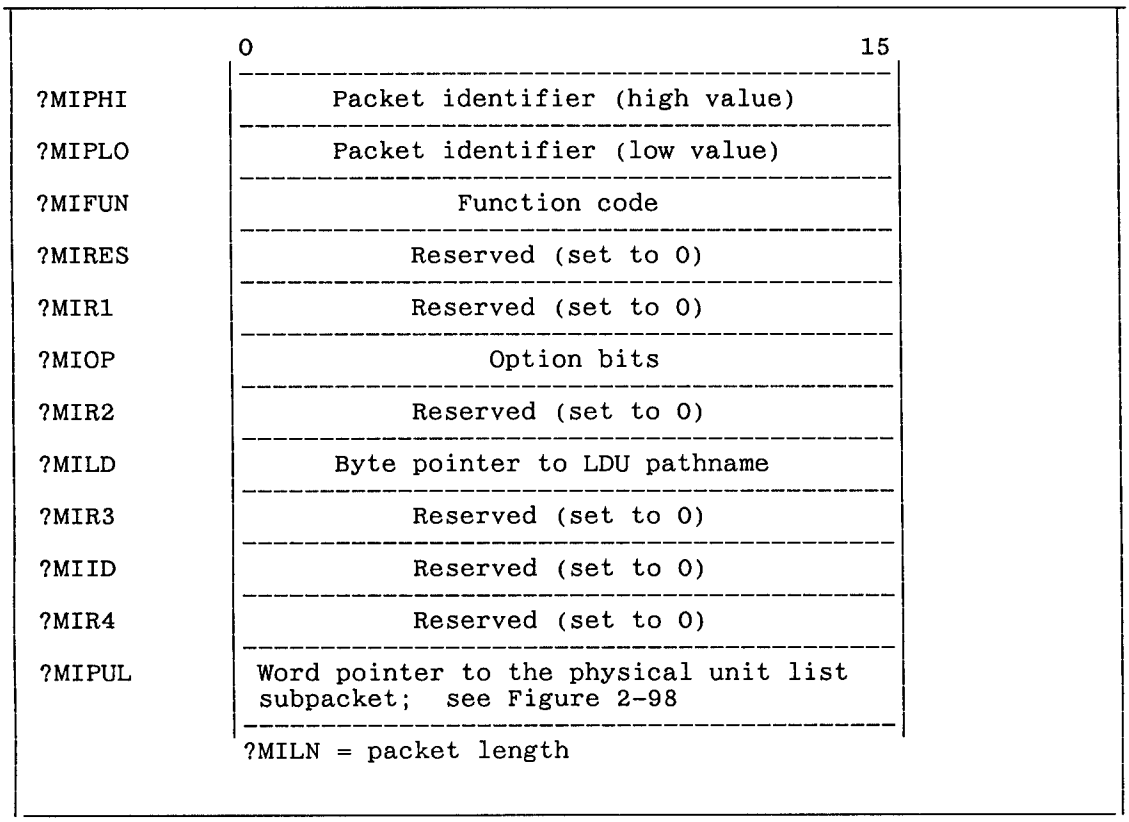


Figure 2-97. Structure of 16-Bit ?MIRROR Packet

Notes

- See the descriptions of ?RELEASE and ?XINIT in this chapter.

?MIRROR Continued

Table 2-74. Contents of 16-Bit ?MIRROR Packet

Offset	Contents
?MIPHI	Packet identifier (high word). Place ?MII1 here.
?MIPLO	Packet identifier (low word). Place ?MII2 here.
?MIFUN	Function code. Place the following value here: - ?MFSYM to synchronize the mirror (i.e., add an image to the mirrored set.)
?MIRES	Reserved. (Set to 0.)
?MIR1	Reserved. (Set to 0.)
?MIOP	Options word. Choose from the following values here, but only when offset ?MIFUN contains ?MFSYM. - ?MBOOP to override the error condition ERMOP. ("Mirrored LDU is out of phase.") - ?MBWAIT to wait for the synchronization process to complete. (The system might return error code ERSFL or ERLDR.)
?MIR2	Reserved. (Set to 0.)
?MILD	Supply a byte pointer to the pathname of the LDU name that the ?MIRROR call addresses. If specified LDU image is not associated with the specified LDU name, the system returns an error.
?MIR3	Reserved. (Set to 0.)
?MIID	Reserved. (Set to 0.)
?MIR4	Reserved. (Set to 0.)
?MIPUL	Supply, a word pointer to the physical unit list subpacket. Figure 2-98 describes this subpacket.

	0 15
?ULPHI	Packet identifier (high word). Place the value of ?ULI2 here.
?ULPLO	Packet identifier (low word). Place the value of ?ULI3 here.
?ULRO	Reserved. (Set to 0.)
?ULNC	Supply the number of the last physical unit that is used; byte pointers to their names appear in the next 16 offsets The number is between 1 and ?MAXNAMES, inclusive.
?ULR1	Reserved. (Set to 0.)
?ULN1	Supply a byte pointer to physical unit name number 1. You must terminate the name with a null byte.
?ULR2	Reserved. (Set to 0.)
?ULN2	Supply a byte pointer to physical unit name number 2. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.
?ULR3	Reserved. (Set to 0.)
?ULN3	Supply a byte pointer to physical unit name number 3. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.
?ULR4	Reserved. (Set to 0.)
?ULN4	Supply a byte pointer to physical unit name number 4. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.
?ULR5	Reserved. (Set to 0.)
?ULN5	Supply a byte pointer to physical unit name number 5. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.
?ULR6	Reserved. (Set to 0.)
?ULN6	Supply a byte pointer to physical unit name number 6. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.
?ULR7	Reserved. (Set to 0.)
?ULN7	Supply a byte pointer to physical unit name number 7. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.
?ULR8	Reserved. (Set to 0.)
?ULN8	Supply a byte pointer to physical unit name number 8. You must terminate the name with a null byte. If you don't supply a byte pointer, supply 0.

?ULLN = packet length

Figure 2-98. Structure of 16-Bit ?MIRROR Subpacket

?MPHIST Starts a histogram on a uni– or multiprocessor system. (32–Bit Processes Only)

AOS/VS only

?MPHIST *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?MPHIST packet

Error Codes in AC0

ERHIS	Error on histogram initialization or deletion
ERMPR	System call parameter address error
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERPRV	Caller not privileged for this action (The caller is not a resident process.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

Use this system call to start a histogram on either a uniprocessor system or on a multiprocessor system.

What It Does

This system call starts a histogram for a range of local addresses in a process. The caller must be a resident process and cannot activate more than one histogram at a time.

If you want information about the calling process, place the value of ?MPH_PID in offset ?MPH_PKT.CODE of the packet and place –1 in offset ?MPH_PKT.PID.

Licensed Material – Property of Data General Corporation

If you want information about another process, place the value of ?MPH_PNAME in offset ?MPH_PKT.PCODE. Either supply the process ID in offset ?MPH_PKT.PID (and 0s in offsets ?MPH_PKT.PNAME and ?MPH_PKT.PSUPPLIED), or supply 0 in offset ?MPH_PKT.PID and the appropriate byte pointer information in offsets ?MPH_PKT.PNAME and ?MPH_PKT.PSUPPLIED.

If you want information about all processes, place the value of ?MPH_PID in offset ?MPH_PKT.PCODE of the packet and place 0 in offset ?MPH_PKT.PID.

Supply the rest of the histogram packet, which includes the starting and ending addresses of the range that ?MPHIST monitors, the size of each interval in the specified range, and the address of the receive buffer for each job processor. If you don't want to specify a receive buffer for a job processor, set this address offset (?MPH_PKT.JP0, ..., ?MPH_PKT.JP15) to 0.

Note that the format of the receive buffer is the same for ?WHIST and ?MPHIST. However, ?MPHIST can have a receive buffer for each job processor.

Setting the interval size to 0 results in a simple histogram that records how often the target process gained CPU control. AOS/VS ignores the starting and ending addresses.

AOS/VS returns histogram statistics in array format to each of your specified receive buffers. The array consists of a fixed-length header followed by double-word entries that correspond to each specified interval.

Side Effects

You can specify receive buffers for job processors that have not been initialized. Why should you? When a job processor is initialized, it can start writing numbers to its receive buffer dynamically. AOS/VS places zeros in the receive buffers at the start of the call regardless of whether the processor has been initialized.

Also, you don't have to specify a receive buffer for every job processor. This is useful if you want a histogram that looks at activity on one job processor only. It will cut down on the amount of space your program uses since you don't have to set aside multiple copies of your receive buffer.

The flexibility of ?MPHIST is such that you decide on the interpretation of the returned data. If you've allocated receive buffers for all job processors and they are homogeneous, then you should be able to average the numbers from all the receive buffers. Such an average yields histogram data for the process on the system. However, if you haven't allocated receive buffers for all the job processors or if not all the job processors are homogeneous, then averaging the numbers gives skewed results. In such a case it would be useful to look at the data on a job processor basis.

Figure 2-99 shows the structure of the ?MPHIST parameter packet, and Table 2-75 describes its contents. Table 2-76 shows the structure and contents of a buffer that receives histogram statistics. See the explanation of ?GHRZ for details about the system clock's frequency.

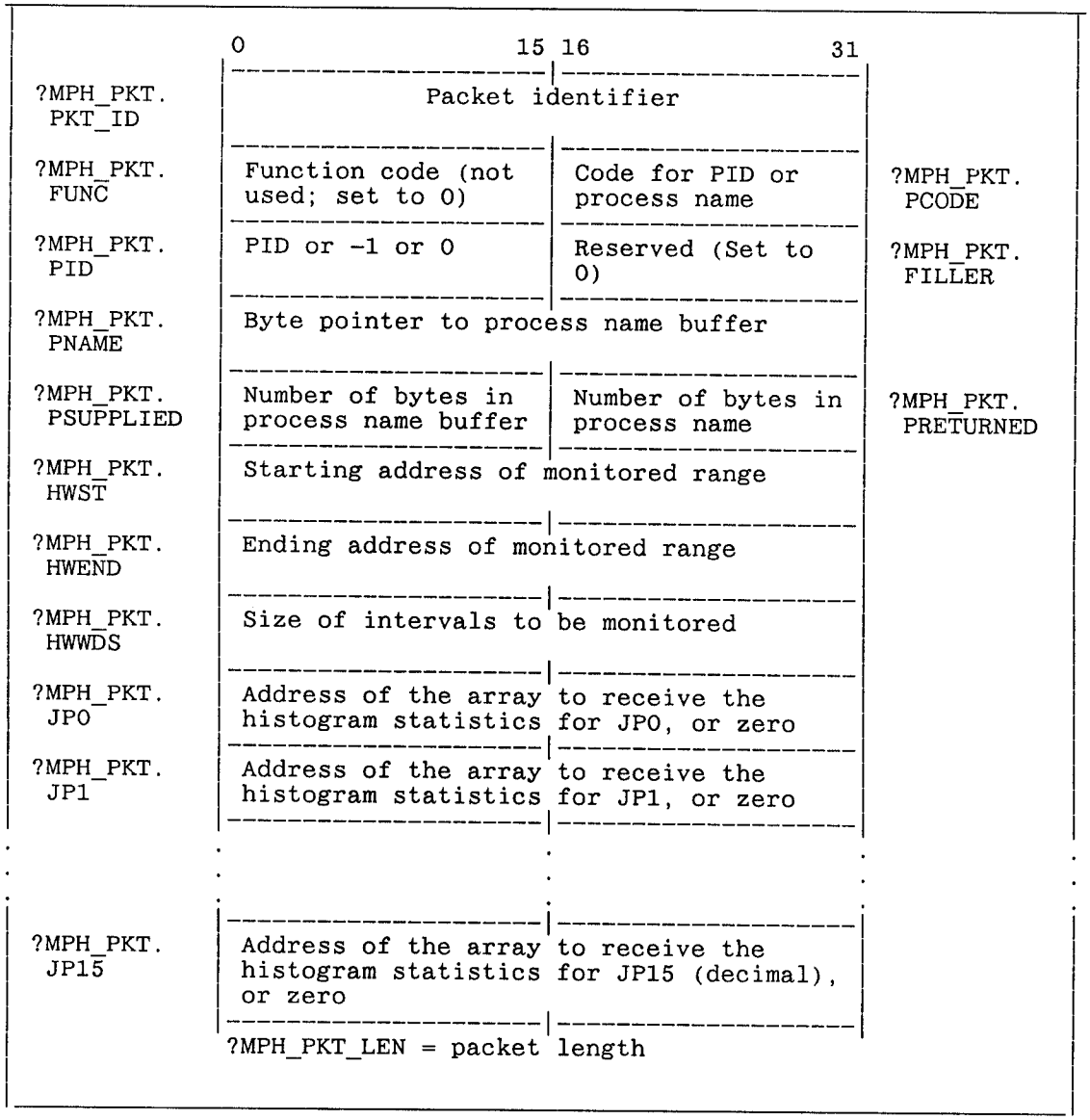


Figure 2-99. Structure of ?MPHIST Packet

Table 2-75. Contents of ?MPHIST Packet

Offset	Contents
?MPH_PKT.PKT_ID (double word)	Packet identifier. Place ?MPHIST_PKT_PKTID here.
?MPH_PKT.FUNC	Function code. Not used. (Set to 0.)
?MPH_PKT.PCODE	Code word into which you place ?MPH_PID when you supply a PID or else ?MPH_PNAME when you supply a process name.
?MPH_PKT.PID	Supply -1 for the calling process. For another process, either place its PID here or else place place 0 here and the process's name information in the two offsets after the next one. For all processes, place 0 here and in the two offsets after the next one.
?MPH_PKT.FILLER	Reserved. (Set to 0.)
?MPH_PKT.PNAME (double word)	If you've placed 0 in offset ?MPH_PKT.PID, then place a byte pointer here pointing to the process's name and place its length in the next offset. Or, place a 0 here for information about all processes.
?MPH_PKT.PSUPPLIED	If you've placed 0 in offset ?MPH_PKT.PID and a byte pointer in the previous offset, then place the number of bytes in the byte pointer buffer here. Or, place a 0 here for information about all processes.
?MPH_PKT.PRETURNED	If you've specified a byte pointer in offset ?MPH_PKT.PNAME, then AOS/VS returns the actual number of bytes in the string in the byte pointer buffer. If you haven't specified a byte pointer in offset ?MPH_PKT.PNAME, place 0 here.
?MPH_PKT.HWST	Supply the starting address of the memory area that you want monitored.
?MPH_PKT.HWEND	Supply the ending address of the memory area that you want monitored.
?MPH_PKT.HWWDS	Supply the number of intervals in the memory area that you want monitored.

(Continues)

Table 2-75. Contents of ?MPHIST Packet

Offset	Contents
?MPH_PKT.JP0	Place zero here if you don't want to receive histogram statistics for job processor 0, or else place the word address here of the array that will receive the histogram statistics.
?MPH_PKT.JP1	Place zero here if you don't want to receive histogram statistics for job processor 1, or else place the word address here of the array that will receive the histogram statistics.
.	.
.	.
.	.
?MPH_PKT.JP15	Place zero here if you don't want to receive histogram statistics for job processor 15, or else place the word address here of the array that will receive the histogram statistics.

(Concluded)

Table 2-76. Structure and Contents of ?MPHIST Histogram Array

	0 15 16 31
?MPHIST_ HTTH	Total number of real-time clock pulses (ticks) counted in this histogram
?MPHIST_ HPRH	Total number of ticks when program counter (PC) was within the target process, but outside the specified range
?MPHIST_ HAPH	Total number of ticks in other processes
?MPHIST_ HSBH	Total number of ticks in AOS/VS, except those recorded when it was in an idle loop
?MPHIST_ HSIH	Total number of ticks in a system idle loop
?MPHIST_ HARAY	Total number of ticks in the first interval
?MPHIST_ HARAY + 2	Total number of ticks in the second interval
.	.
.	.
.	.
?MPHIST_ HARAY + 2*(n-1)	Total number of ticks in the nth interval

?MYTID Gets the priority and TID of the calling task.

?MYTID
error return
normal return

Input

None

Output

AC0	TID of the calling task
AC1	Priority level of the calling task
AC2	Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?MYTID allows you to identify the calling task by its priority and TID. ?MYTID can be a useful preliminary to other tasking system calls that require TIDs and/or priority levels as input parameters.

What It Does

?MYTID returns the TID of the calling task to AC0 and its priority level to AC1.

?ODIS

Disables console interrupts.

?ODIS
error return
normal return

Input

None

Output

None

Error Codes in AC0

No error codes are currently defined.

Why Use It?

By default, the operating system enables console interrupts when each program begins to execute. ?ODIS lets you override this default or lets you revoke an interrupt enable that was caused by an ?OEBL, ?INTWT, or ?CHAIN.

What It Does

?ODIS disables all console interrupts (CTRL-C CTRL-A sequences) until one of the following occurs:

- You explicitly re-enable CTRL-C CTRL-A sequences by issuing ?OEBL.
- You issue ?INTWT to define a console interrupt task (a task to monitor the process console for CTRL-C CTRL-A sequences).
- You issue ?CHAIN to chain to a new program.

Notes

- See the descriptions of ?OEBL, ?INTWT, and ?CHAIN in this chapter.

?OEBL — Enables console interrupts.

?OEBL
error return
normal return

Input

None

Output

None

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?OEBL to revoke a previous ?ODIS or to re-enable console interrupts after an ?INTWT sequence.

What It Does

?OEBL is the inverse of ?ODIS; that is, it re-enables console interrupts (CTRL-C CTRL-A sequences). Note that ?OEBL does not define a task for processing console interrupts. (To do this, issue ?INTWT.)

Notes

- See the descriptions of ?ODIS, ?INTWT, and ?CHAIN in this chapter.

?OPEN

Opens a file.

?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
ERMPR	System call parameter address error
ERVWP	Invalid word pointer passed as a system call argument

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.

Figure 2-100 shows the structure of the ?OPEN packet, and Table 2-77 describes each offset and bit position in the packet. The accompanying text explains the options you can exercise in setting the packet specifications.

Figure 2-100 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. The only extension that applies to ?OPEN is the labeled magnetic tape extension. (For information on the other extensions, see the descriptions of ?READ and ?WRITE.) 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 write purposes only.
- You must set all unused bits in offset ?ISTI to 0. (To do this, set the bits you want to use.)
- 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.

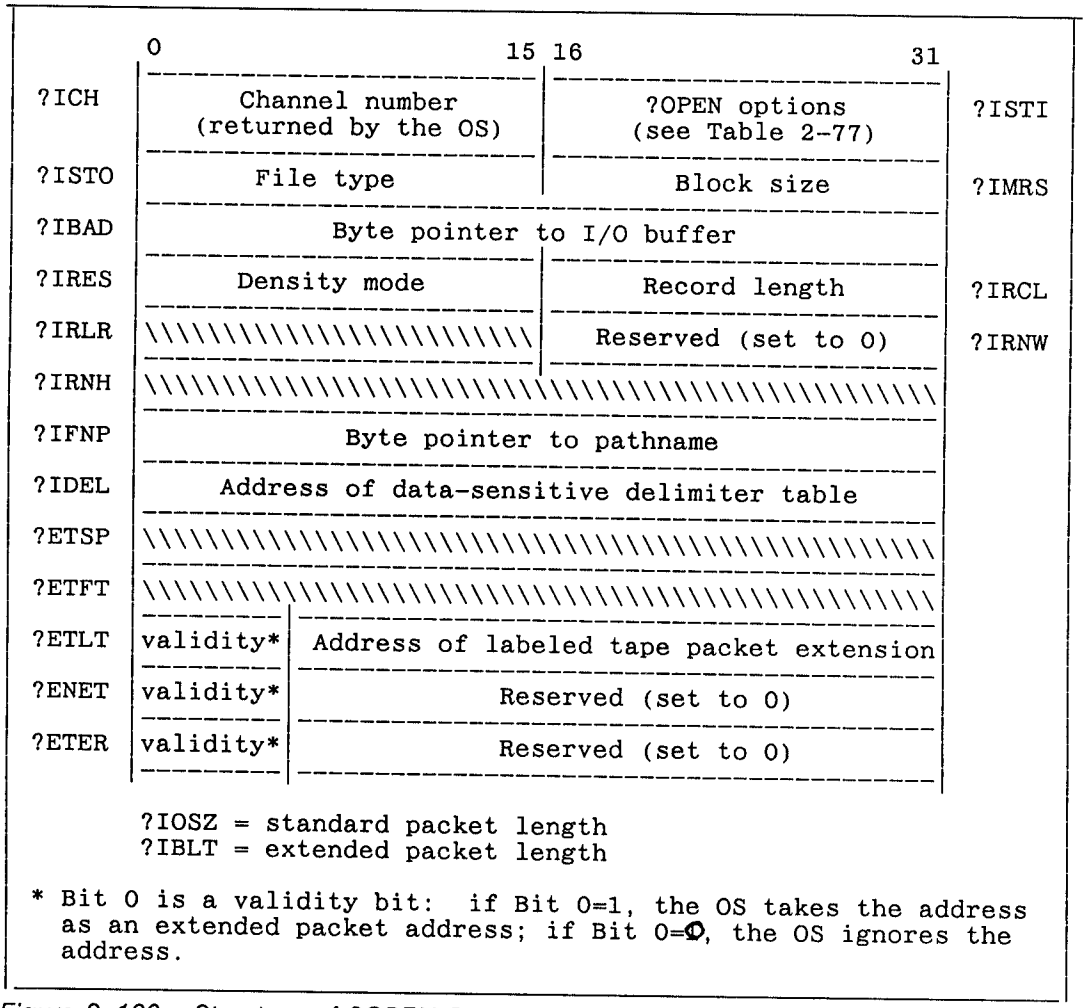


Figure 2-100. Structure of ?OPEN Packet

Table 2-77. Contents of ?OPEN Packet*

Offset	Contents
?ICH	Channel number (assigned by the O S).
?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 (?READ/?WRITE only). 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).

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

?OPEN Continued

Table 2-77. Contents of ?OPEN Packet*

Offset	Contents
?ISTI	Priority read ?PDEL--Open file for priority read. DEFAULT = 0 (file has normal priority). Creation option ?OFCE--Create the file, return unless it already exists; if it already exists, ignore. ?OFCR--Create the file; then open it. ?OFCR!?OFCE--Delete the file; then recreate it and open it. DEFAULT = 0 (if file does not exist, do not create it; return ?OPEN error). Input/output ?OFIN--Open for input. ?OFOT--Open for output. ?OFIO--Open for input/output. Record format ?RTDY--Dynamic-length. ?RTDS--Data-sensitive. ?RTFX--Fixed-length. ?RTVR--Variable-length. ?RTUN--Undefined-length. ?RTVB--Variable block, variable record.
?ISTO	Left byte ?SHOP--Shared ?OPEN request. ?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).

* There is no default unless otherwise specified.

(Continued)

Table 2-77. Contents of ?OPEN Packet*

Offset	Contents
?ISTO	Right byte File Type. (See Table 2-79.) 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).
?IMRS	Physical block size (in bytes). DEFAULT = -1 (block size = 2 Kbytes)
?IBAD (double word)	Byte pointer to record I/O buffer. DEFAULT = -1 (deferred until you issue ?READ or ?WRITE).
?IRES	Density mode (for magnetic tapes only) ?IDAM--Automatic select. ?ID8--Density 800 bytes/inch. ?ID16--Density 1600 bytes/inch. ?ID62--Density 6250 bytes/inch. DEFAULT = 0 (use density mode specified during the system-generation procedure). Labeled tape format ?OIBM--Open as an IBM labeled tape. ?OANS--Open as an ANSI labeled tape. DEFAULT = 0 (open as an AOS/VS or AOS/DVS tape).
?IRCL	Record length. DEFAULT = -1 (deferred until you issue ?READ or ?WRITE).

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

Licensed Material - Property of Data General Corporation

Table 2-77. Contents of ?OPEN Packet*

Offset	Contents
?IRLR	Number of bytes transferred (?READ/?WRITE only).
?IRNW	Reserved. (Set to 0.)
?IRNH (double word)	Record Number (?READ/?WRITE only).
?IFNP (double word)	Byte pointer to pathname.
?IDEL (double word)	Address of the delimiter table. DEFAULT = -1 (use default delimiters: null, NEW LINE, form feed, and carriage return).
?ETLT (double word)	Address of the labeled magnetic tape packet extension.
?ENET (double word)	Reserved. (Set to 0.)
?ETER (double word)	Reserved. (Set to 0.)

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

The File Specifications Word

As Table 2-77 indicates, you can select a number of optional file specifications in offset ?ISTI.

To select a record format different from the one specified in the ?CREATE packet, select the ?ICRF mask in offset ?ISTI and one of the record format masks. The record format masks correspond to the operating system record types: dynamic-length, fixed-length, data-sensitive, variable-length, and undefined-length. For example, to define the record type as data-sensitive, specify the following:

?ICRF!?RTDS

The Append Field option (?APND) moves the file pointer to the end of the file. If you select this option and then write to the file, the operating system appends the new material to the end of the file.

The exclusive open option (?IEXO) opens the file for the exclusive use of the calling task. No other task, either in the calling process or in any other process, can open the file until the current caller closes it. If you do not specify this option, you can open a file for more than one task or process to use at the same time, as long as you supply an I/O packet for each channel. (The operating system assigns a different channel number to the file on the second and subsequent ?OPENs.)

You can open a file that the operating system peripheral manager (PMGR) controls, and give its channel priority read status by selecting mask ?PDEL in offset ?ISTI. When a channel has priority read status, the operating system places that channel's ?READ requests ahead of the ?READ requests from all other channels and executes the priority reads first.

The ?OPEN packet allows you to simultaneously create and open files. Table 2–78 describes each of the file creation options for offset ?ISTI.

Table 2–78. File Creation Options for Offset ?ISTI

Mask	System Directive
?OFCE	Create the file, unless it already exists. (Ignore ?OFCE if the file already exists.)
?OFCR	Create the file; if it already exists, return an error to AC0, and pass control to the error return.
?OFCE!?OFCR	If the file already exists, delete it, and then recreate it. (You cannot use this option for IPC files.)

Table 2–79 lists the more common file types you can create with ?OPEN. Table 2–11, under ?CREATE, lists all the file types you can create with ?OPEN.

Offset ?ISTI also contains an input/output field with three options:

- ?OFIN, which opens the file for input (?READs).
- ?OFOT, which opens the file for output (?WRITEs).
- ?OFIO, which opens the file for input and/or output (?READs and/or ?WRITEs).

You must select one of these options to perform I/O on the file. Otherwise, ?OPEN succeeds, but the subsequent ?READ or ?WRITE fails.

Table 2-79. Common File Types You Can Create with ?OPEN

File Type	Meaning	Comments
?FUDF	User Data File	The OS creates this type when you default the file type option (set right byte of ?ISTO to 0).
?FTXT	Text File	Should contain ASCII code.
?FPRV	32-bit Program File	An executable 32-bit program file; should contain linked, executable code.
?FDIR	Disk Directory	If you use ?OPEN to create a file of this type, you can default only the following: hash frame size, maximum number of index levels, and ACL.
?FIPC	IPC File	Directs the OS to create an IPC file or open an existing IPC file to allow full-duplex communications between two processes.
?FCPD	Control Point Directory	Although you can use ?OPEN to create a CPD, we recommend that you use ?CREATE instead.

Offset ?ISTO

The right byte of offset ?ISTO specifies the type of file you are opening. Use this field only if you are creating the file with ?OPEN. If the file you are opening already exists, the operating system ignores your input to this field and fills it with the file type you specified when you created the file.

You can set the ?SHOP parameter in offset ?ISTO to open the file for shared access. The ?SHOP parameter directs the operating system to read the file that you specify into one or more shared pages for access by more than one process. When you set the ?SHOP parameter, the operating system automatically performs an ?SOPEN on the file, and then initiates the standard ?SPAGE and ?RPAGE operations.

Note that you cannot set the ?SHOP parameter if you are creating or recreating the file with this ?OPEN. Also, note that if you open the file with a standard ?OPEN, the shared file facility does not protect your file from modifications.

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.

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 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 2-101 depicts a sample delimiter table with bits set to make the null, carriage return (015 octal), and rubout (177 octal) characters data-sensitive delimiters.

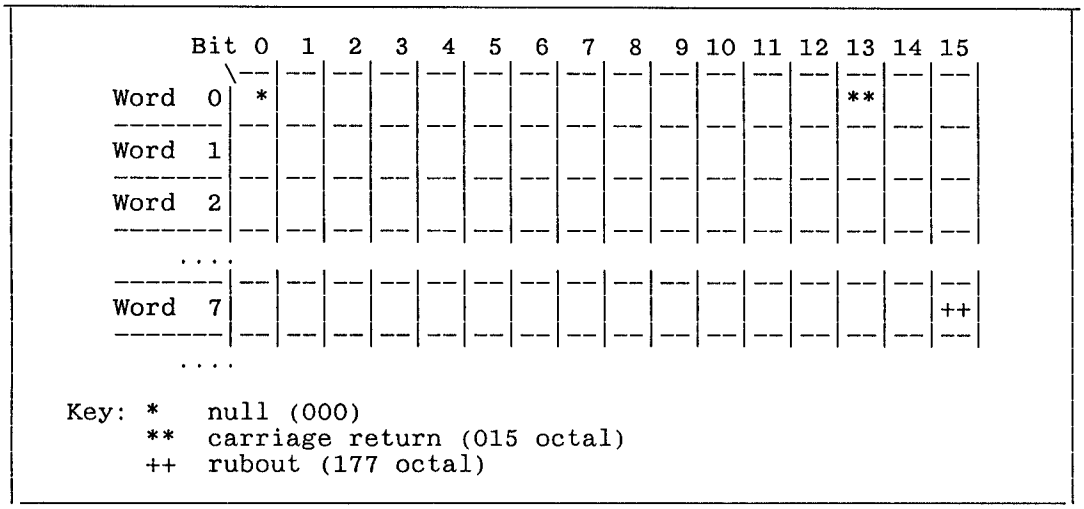


Figure 2-101. Sample Delimiter Table

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. You supply the packet as an extension to the packet for a ?OPEN, ?READ, or ?WRITE system call. Figure 2-102 contains the extension packet. Table 2-80 describes its contents.

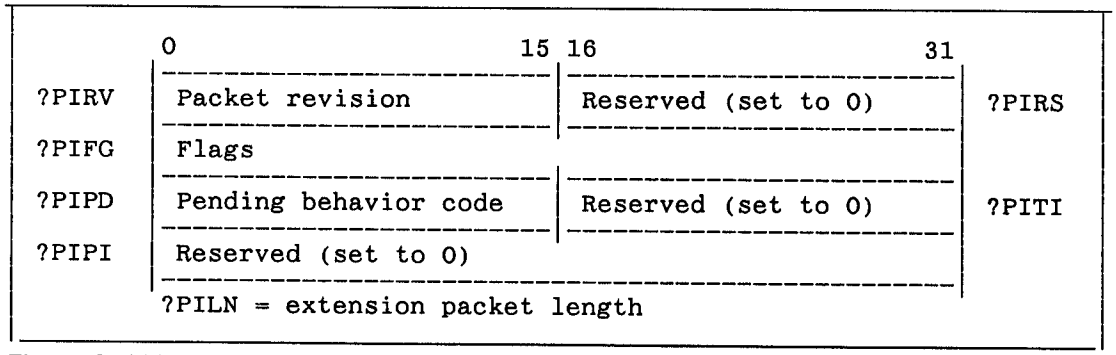


Figure 2-102. Structure of ?OPEN Extension Packet for Pipes

Licensed Material - Property of Data General Corporation

Table 2-80. Contents of ?OPEN Extension Packet for Pipes

Offset	Contents
?PIRV	Packet revision number. Place the value of symbol ?PKR1 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. ?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 type MTB controller, you can set offset ?IRES to a specific density mode for the tape. There are three 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 unit to a density mode of 1600 bytes/inch.
- ?ID62 sets the unit to a density mode of 6250 bytes/inch.

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 2-103 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 2-81.

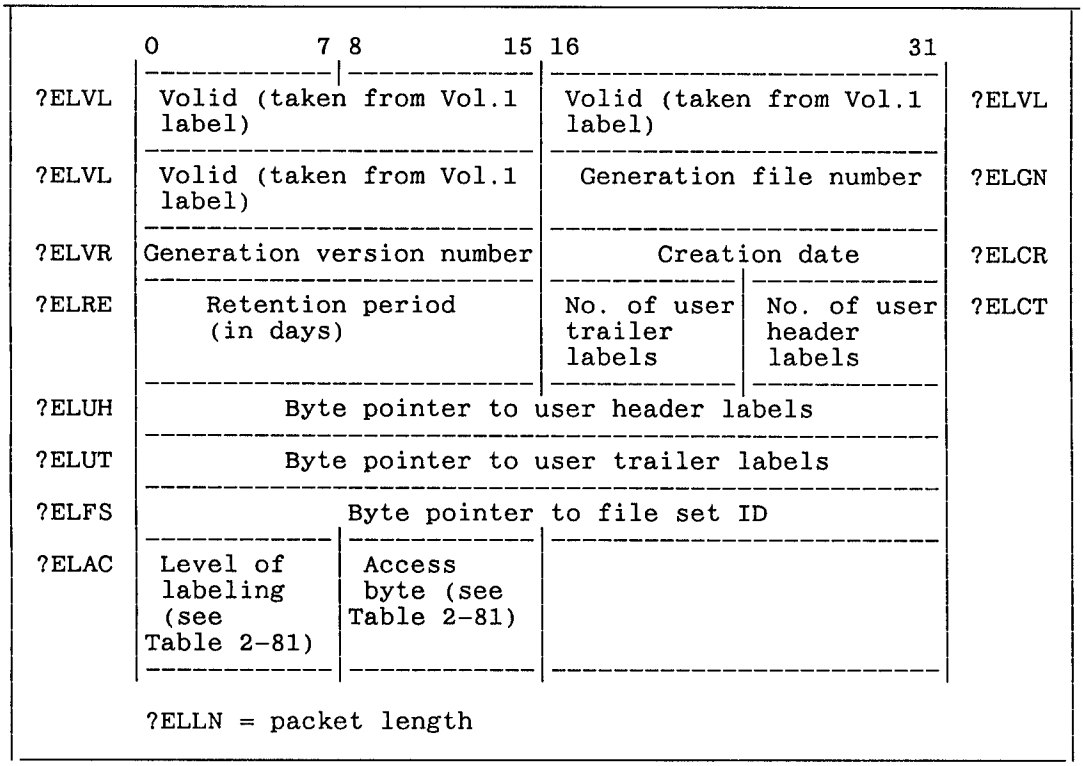


Figure 2-103. Structure of Labeled Magnetic Tape Packet Extension

Table 2-81. Contents of Labeled Magnetic Tape Packet Extension*

Offset	Contents
?ELVL	Volid (returned by the OS) taken from Volume 1 Label 3 words.
?ELGN	Generation number of file (from 0 through 9999). DEFAULT = -1 (if you specify -1, the OS uses 0001 when it writes to the file, and does not check or return the generation number when it reads the file).
?ELVR	Generation version number (from 0 through 99). DEFAULT = -1 (if you specify -1, the OS uses 00 when it writes to the file, and does not check or return the version number when it reads the file).
?ELCR	Creation date in standard form: dd.mm.yy. DEFAULT = current system date.
?ELRE	Retention period in days. DEFAULT = 90 days.
?ELCT	Right byte: number of user header labels, within the range 1 through 9 (0 means none). Left byte: number of user trailer labels, within the range 1 through 9 (0 means none). At ?OPEN time, the OS either reads or writes the header labels, depending on whether you opened the file for input (reading) or for output (writing). When you close the labeled tape file, the OS writes the user trailer labels from the area specified in offset ?ELUT. If the file was not modified, the OS reads the trailer labels into the area specified in ?ELUT.

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

Table 2-81. Contents of Labeled Magnetic Tape Packet Extension*

Offset	Contents
?ELUH (double word)	Byte pointer to an area containing the header labels; all labels are contiguous and sequential.
?ELUT (double word)	Byte pointer to an area containing the trailer labels; all labels are contiguous and sequential.
?ELFS (double word)	Byte pointer to file set ID; if you set ?ELFS to -1, the OS ignores the file set ID when it reads, and uses the default when it writes. DEFAULT = 0 (Use volid for first volume in file set.)
?ELAC	Level of labeling (left byte) ?ELL1--Level 1 (ANSI or IBM). ?ELL2--Level 2 (ANSI or IBM). ?ELL3--Level 3 (ANSI). DEFAULT = 0 (if ANSI format, ANSI Level 3; if IBM format, IBM Level 2). Access byte (right byte): For ANSI format, set this byte to one or more characters that specify access to the tape. (For IBM format, set this byte to 0.) DEFAULT = 0 (if ANSI format, blank space; if IBM format, 0).

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

Note that there is no default value for the tape volume identifier (volid). If you want to create and open a labeled tape file and default the labeled tape extension, you must first establish a volid for the tape volume. You can do this with the ?LABEL system call or with the CLI LABEL utility.

You can create files on any labeled tape volume as long as you specify the correct volid. If the file does not exist, the operating system creates it after the last file.

If the file exists, but its retention period has expired or you selected the delete and create option in the ?OPEN packet, the operating system overwrites the existing file with the new file. Note that the delete and create option also deletes all subsequent files on the tape, because magnetic tape files are sequential.

When you open a labeled magnetic tape, do not use the ?OPEN option ?OFIO in offset ?ISTI, which opens the file for both reading and writing. If you use this option, the ?OPEN fails and the operating system returns error code ERIOO (illegal option for open type) in AC0.

?OPEN Continued

You can use as many as nine user header labels and nine user trailer labels. Each user header and trailer label is 80 bytes (characters) long, where the last 76 bytes are user data. If the operating system finds a null character when writing the label, it takes the null as a terminator, and if necessary, pads the label with spaces to fill out the 80 character positions. The next label starts 80 bytes past the first label. When the operating system reads the label, it returns the data from the first label to the first 76 bytes of the ?ELUT area, the second label to the second 76 bytes, and so forth.

If you intend to write to the labeled tape file, be sure to set the following fields in the file's ?OPEN packet:

- Record format in offset ?ISTI (with the flag ?ICRF; for example, ?RTFX!?ICRF for fixed-length records).
- Record length in offset ?IRCL.
- Block length in offset ?IMRS.

These fields must contain valid information; otherwise, another operating system may not be able to read the tape.

When the operating system writes to a labeled tape file, it will overwrite the existing file data and all subsequent data on the tape, if the retention period (?ELRE) has expired. The default retention period is 90 days.

If you want to read a file on the labeled tape, do not set ?ICRF when you set the record format, because the operating system will be unable to return the record format to ?ISTI after it reads the file.

The operating system returns the file's record length to ?IRCL after it reads a labeled magnetic tape file, and returns the block length to ?IMRS. If you want the operating system to return the block length after a ?READ, be sure to set ?IMRS to -1 in the ?OPEN packet.

Sample Packet

The following sample packet opens a file with data-sensitive records for file I/O. The file already exists:

```
PKT:    .BLK    ?IBLT           ;Allocate enough space for packet.
                                ;Packet length = ?IBLT.
        .LOC    PKT+?ICH        ;Channel number.
        .WORD   0               ;The OS returns the channel number.
        .LOC    PKT+?ISTI       ;?OPEN options.
        ?ICRF!?RTDS!?OFIO       ;Change record format to that
                                ;specified in record format field
                                ;(?ICRF), which is data-sensitive
                                ;(?RTDS), and open for I/O (?OFIO).
        .LOC    PKT+?ISTO       ;File type.  The OS assumes the file
        .WORD   0               ;type that you specified when you
                                ;created the file.
        .LOC    PKT+?IMRS       ;Physical block size is
        .WORD   -1              ;2 Kbytes (-1 is the default).
        .LOC    PKT+?IBAD       ;Specify byte pointer to record I/O
                                ;buffer.
        .DWORD  BUF*2           ;Byte pointer to BUF.
        .LOC    PKT+?IRES       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?IRCL       ;Specify record length.
        .WORD   120.            ;Maximum record length is 120.
                                ;characters.
        .LOC    PKT+?IFNP       ;Specify byte pointer to pathname.
        .DWORD  PTH*2           ;Byte pointer to PTH.
        .LOC    PKT+?IDEL       ;Specify delimiter table address.
        .DWORD  -1              ;Use default delimiters: NEW LINE,
                                ;form feed, and carriage return (-1
                                ;is the default).
         .LOC    PKT+?IBLT      ;End of packet.
```

The following sample packet deletes a file with fixed-length records, recreates it, and then opens it for output:

```
PKT:    .BLK    ?IBLT           ;Allocate enough space for packet.
                                ;Packet length = ?IBLT.
        .LOC    PKT+?ICH        ;Channel number.
        .WORD   0               ;The OS returns the channel number.
        .LOC    PKT+?ISTI
        .WORD   ?ICRF!?RTFX!?OFCR!?OFCE!?OFOT   ;Change record format
                                ;to that specified in record format
                                ;field (?ICRF), which is fixed-length
                                ;(?RTFX), delete, and then recreate
                                ;the file (?OFCR!?OFCE), and open the
                                ;file for output (?OFOT).
        .LOC    PKT+?ISTO       ;Specify file type.  The OS assumes
        .WORD   0               ;that it is a user data file (?FUDF)
                                ;by default.
        .LOC    PKT+?IMRS       ;Physical block size is
        .WORD   -1              ;2 Kbytes (-1 is the default).
        .LOC    PKT+?IBAD       ;Specify byte pointer to record I/O
                                ;buffer.
        .DWORD  BUF*2           ;Byte pointer to BUF.
```

?OPEN Continued

```
        .LOC    PKT+?IRES       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?IRCL       ;Specify record length. (?READ and
                                ;?WRITE use this value.)
        .WORD   33.             ;Maximum record length is 33-byte
                                ;fixed records.
      .LOC    PKT+?IRLR         ;Record length.  (?READ and ?WRITE
                                ;use this value.)
      .WORD   0                 ;The OS returns this value.
      .LOC    PKT+?IFNP         ;Specify byte pointer to pathname.
      .DWORD  PTH*2             ;Byte pointer to PTH.
      .LOC    PKT+?IDEL         ;Specify delimiter table address.
      .DWORD  -1                ;Use default delimiters: NEW LINE,
                                ;form feed, and carriage return
                                ;(-1 is the default).
      .LOC    PKT+?IBLT         ;End of packet.
```

Notes

- See the descriptions of ?CLOSE, ?CREATE, ?READ, ?WRITE, ?SPAGE, ?RPAGE, ?SDLM, and ?LABEL in this chapter.
- See the description of ?CGNAM in this chapter to get a complete pathname from a channel number.

?OPER Creates and maintains an operator interface.

?OPER *[packet address]*
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	None
Error Codes	None
Parameter Packet	Some

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?OPER packet

Error Codes in AC0

ERBTL	Buffer too long
ERDAE	Operator already exists
ERDEB	Daemon resource error in error buffer
ERDNE	Task is not an operator
ERDRF	Daemon resource failure
ERIRB	Insufficient room in buffer
ERNOA	No operator available
ERNRQ	No matching operator request
ERPKT	Invalid packet identifier
ERPNO	Process is not an operator
ERPRE	Invalid system call parameter
ERVWP	Invalid word pointer passed as system call argument

Why Use It?

Use this system call to communicate with an operator and receive its responses.

What It Does

This system call performs six basic functions. They are

- Operator on.
- Operator off.
- Send operator request.
- Operator receive request.
- Operator respond to request .
- Operator information.

You supply a main packet in all operator interface requests. You also supply a subpacket that the main packet points to. This subpacket contains the details about one of the six functions mentioned above.

Figure 2-104 shows the structure of the ?OPER main packet, and Table 2-82 describes its contents. After Table 2-82 the subpackets are described in the following order.

Function Code	Function Description	Structure of Subpacket	Contents of Subpacket
?OPON	Operator on	Figure 2-105	Table 2-83
?OPOFF	Operator off	Figure 2-106	Table 2-84
?OPSEND	Send operator request	Figure 2-107	Table 2-85
?OPRCV	Operator receive request	Figure 2-108	Table 2-86
?OPRESP	Operator respond to request	Figure 2-109	Table 2-87
?OPINFO	Operator information	Figure 2-110	Table 2-88

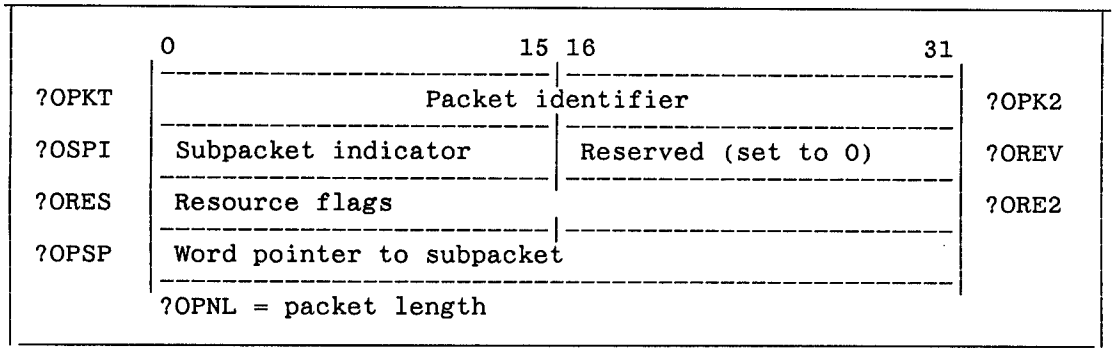


Figure 2-104. Structure of ?OPER Main Packet

Table 2-82. Contents of ?OPER Main Packet

Offset	Contents
?OPKT/ ?OPK2	Packet identifier. Place ?OPID here.
?OSPI	Function code. It specifies the operation you want, and indicates the existence of a unique subpacket to the operating system. You must supply this main packet and exactly one subpacket for each issuance of ?OPER. Select from the following values: ?OPON -- Become an operator daemon. ?OPOFF -- Resign as an operator daemon. ?OPINFO -- Request information about the current operator environment. ?OPSEND -- Issue an operator request. ?OPRCV -- Receive an operator request. ?OPRESP -- Respond to an operator request.
?OREV	Reserved; set to 0.
?ORES/ ?ORE2	Resource indicator bit flags. This double word contains device bit indicators that represent the various system resources. Each indicator corresponds to the resource(s) the function is intended for. The bit numbers and their resources are as follows: ?OBLD -- Labeled diskettes (LD). ?OBUD -- Unlabeled diskettes (UD). ?OBLT -- Labeled tapes (LT). ?OBUT -- Unlabeled tapes (UT). ?OBCO -- Consoles (C). ?OBPR -- Printers (P). ?OBBQ -- Batch queues (B). ?OBHD -- Hard disks (HD). ?OBQU -- Queues (Q). ?OBCD -- Other cooperative devices (OCD). ?OBGM -- Generic media (GM). ?OBMI -- Miscellaneous (M). For example, to specify consoles and printers, supply the symbol 1S?OBCO!1S?OBPR. Looking at PARU.32 tells you that an equivalent symbol is ?OPCO!?OPPR.
?OPSP	Word pointer to subpacket. Supply a word pointer to the subpacket for the function you selected in offset ?OPKT/?OPK2.

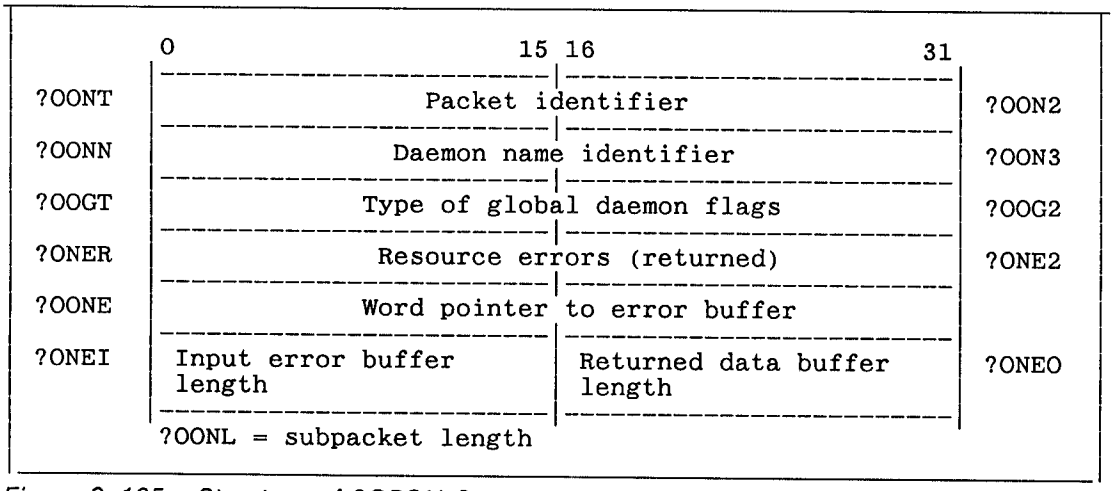


Figure 2–105. Structure of ?OPON Subpacket

Table 2-83. Contents of ?OPON Subpacket

Offset	Contents
?OONT/ ?OON2	Packet identifier. Place ?ONID here.
?OONN/ ?OON3	Daemon name. This offset contains the name that uniquely identifies the global daemon(s) being established for the resources that offset ?ORES in the main packet indicates. ?OONN/?OON3 is 32 bits long; you must supply a value here when you declare any type of a global daemon. Set this field to zero when you are making a request for declaring a local daemon.
?OOGT/ ?OOG2	Flags word. This double word identifies the type of global daemon that you want to become for each resource that offset ?ORES of the main packet specifies. When you want to become an exclusive operator daemon for a resource, set the bit indicator flag corresponding to the resource bit flag in offset ?ORES. Doing so declares exclusive operator global-type daemons for specified resources simultaneously with requests to declare global operator-type daemons for other resources. If you want to become a local daemon, then set this field to zero.
?ONER/ ?ONE2	Resource error. The operating system returns the success or failure of your becoming a daemon for the specified resources in offset ?ORES/?ORE2 of the main packet. If you are unsuccessful, then the operating system sets the device bit indicator corresponding to the failed resource in ?ORES/?ORE2. If you are successful for all resources requested, then the operating system sets offset ?ONER/?ONE2 to zero.
?OONE (double word)	Error buffer pointer. Supply the word address of the error data buffer. The operating system returns error codes to this buffer that indicate why it did not declare the daemon for the indicated resources in offset ?ONER/?ONE2 above. Each system resource has a dedicated word entry in the buffer to hold the encountered error codes. If the operating system detects an error while declaring a daemon for a particular resource, it loads the appropriate error code into the correct entry. If you don't want particular errors, then set this field to zero before issuing ?OPER. Here is the resource error buffer word entry format: 1. error code for labeled diskettes 2. error code for unlabeled diskettes 3. error code for labeled tapes 4. error code for unlabeled tapes 5. error code for consoles 6. error code for printers 7. error code for batch queues 8. error code for hard disks 9. error code for queues 10. error code for other cooperative devices 11. error code for generic media 12. error code for miscellaneous

(continues)

Table 2-83. Contents of ?OPON Subpacket

Offset	Contents
?ONEI	Error buffer length. Supply the word length of the error buffer. The buffer must be large enough to accommodate all of the defined system resources, so place at least 12 here. If offset ?OONE contains zero, the system ignores this word.
?ONEO	Returned error buffer length. The system contains the word length of the data buffer returned. The system ignores this offset if offset ?OONE contains zero.

(concluded)

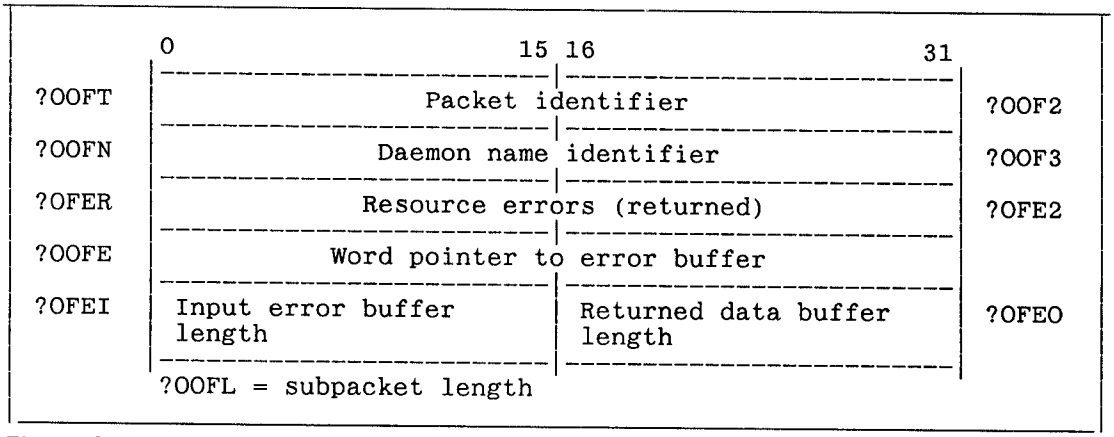


Figure 2-106. Structure of ?OPOFF Subpacket

Table 2-84. Contents of ?OPOFF Subpacket

Offset	Contents
?00FT/ ?00F2	Packet identifier. Place ?OFID here.
?00FN/ ?00F3	Daemon name. This offset contains the same name identifier for each global daemon that is requesting to resign. The identifier is limited to 32 bits; it is the same identifier that was set in the ?OPON function. Set this field to zero when the daemon that wants to resign is a local daemon.
?OFER/ ?OFE2	Resource error. The operating system returns the success or failure of resigning a daemon for the specified resources in offset ?ORES/?ORE2 of the main packet. If you are unsuccessful, then the operating system sets the device bit indicator corresponding to the failed resource in ?ORES/?ORE2. If you are successful for all resources requested, then the operating system sets offset ?OFER/?OFE2 to zero.
?00FE (double word)	Error buffer pointer. Supply the word address of the error data buffer. The operating system returns error codes to this buffer that indicate why it did not resign the daemon for the indicated resources in offset ?OFER/?OFE2 above. Each system resource has a dedicated word entry in the buffer to hold the encountered error codes. If the operating system detects an error while resigning a daemon for a particular resource, it loads the appropriate error code into the correct entry. If you don't want particular errors codes, then set this field to zero before issuing ?OPER. Here is the resource error buffer word entry format: 1. error code for labeled diskettes. 2. error code for unlabeled diskettes. 3. error code for labeled tapes. 4. error code for unlabeled tapes. 5. error code for consoles. 6. error code for printers. 7. error code for batch queues. 8. error code for hard disks. 9. error code for queues. 10. error code for other cooperative devices. 11. error code for generic media. 12. error code for miscellaneous.
?OFEI	Error buffer length. Supply the word length of the error buffer. The buffer must be large enough to accommodate all of the defined system resources, so place at least 12 here. If offset ?00FE contains zero, the system ignores this word.
?OFEO	Returned error buffer length. The system contains the word length of the data buffer returned. The system ignores this offset of offset ?00FE contains zero.

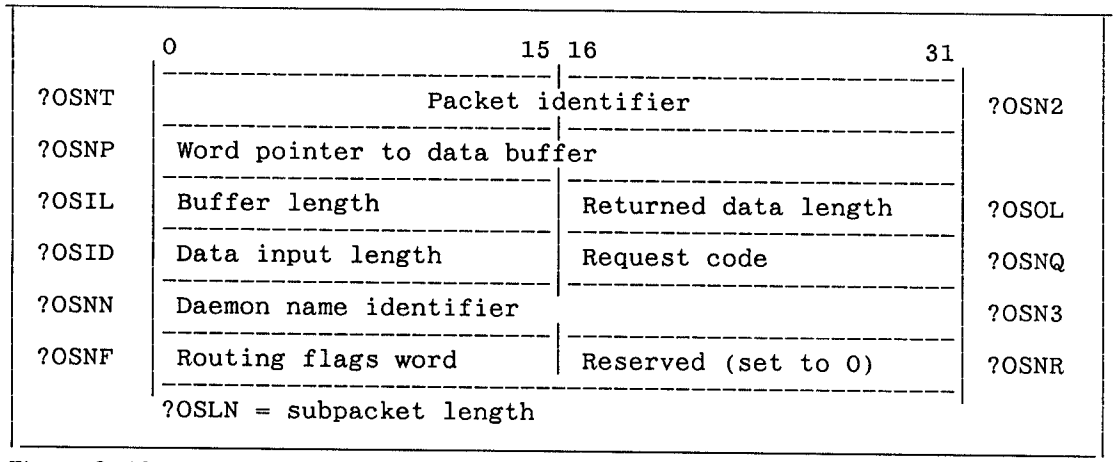


Figure 2-107. Structure of ?OPSEND Subpacket

Table 2-85. Contents of ?OPSEND Subpacket

Offset	Contents
?OSNT/ ?OSN2	Packet identifier. Place ?SEID here.
?OSNP (double word)	Data buffer pointer. This offset contains the word address of the data buffer. The buffer transfers data that is associated with particular operator requests. The operator daemon specifies the format of the data.
?OSIL	Supply the word length of the data buffer.
?OSOL	The operating system returns the word length of data returned to you in the buffer.
?OSID	Supply the number of words of input data in the buffer.
?OSNQ	Supply the request code for the function you are sending to the local daemon. Here is a list of the codes with their resources and functional descriptions: ?ORLO (LD) -- labeled diskette open. ?ORLC (LD) -- labeled diskette close. ?ORMNV (LD) -- mount next volume. ?ORMU (LD) -- mount error.
?OSNN/ ?OSN3	Daemon identifier. For sending requests to local daemons, set this offset to zero and place ?OSNL in offset ?OSNF. For sending requests to a specific global daemon, supply the name identifier of the global daemon and place ?OSNG in offset ?OSNF. For sending requests to any available global daemon, set this offset to zero and place ?OSNO in offset ?OSNF. The operating system returns the name identifier of a global daemon. You could then use this name identifier in subsequent ?OPSEND requests.
?OSNF	Flags word. This offset contains flag bits with routing information for the ?OPSEND function. Set the flag bit that corresponds to the routing schemes as follows: ?OSNL -- send to the local daemon for the resource in offset ?OSNQ. ?OSNG -- send to a specific global daemon, whose name is in offset ?OSNN/?OSN3, for the resource in offset ?OSRE of the main packet. ?OSNO -- send to any available global daemon for the resource; the operating system returns, in offset ?OSNN/?OSN3, the name identifier of the global daemon where it routed the resource.
?OSNR	Reserved; set to 0.

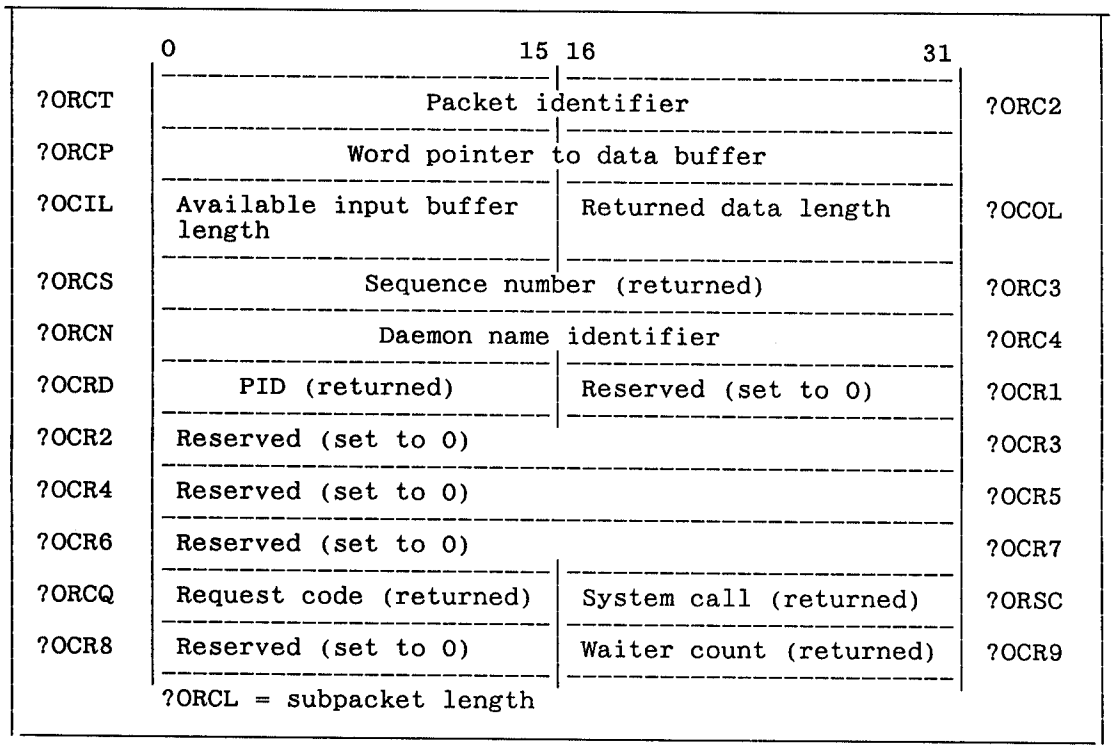


Figure 2–108. Structure of ?OPRCV Subpacket

Table 2-86. Contents of ?OPRCV Subpacket

Offset	Contents
?ORCT/ ?ORC2	Packet identifier. Place ?RCID here.
?ORCP (double word)	Data buffer pointer. This offset contains the word address of the data buffer. The buffer transfers data that is associated with particular operator requests. The operator daemon specifies the format of the data.
?OCIL	Input buffer length. This offset contains the word length of the input data buffer.
?OCOL	Output buffer length. This offset contains the word length of data returned to you in the buffer.
?ORCS/ ?ORC3	Sequence number. The operating system returns a unique sequence identifying number. The number is used to match responses with requests from callers.
?ORCN/ ?ORC4	Daemon name. Supply the name identifier of the calling daemon. If it is a global daemon, then supply the associated name identifier. If it is a local daemon, then supply zero.
?OCRD	PID. The operating system returns the PID of the sender process.
?OCR1	Reserved. (Set to 0.)
?OCR2/ ?OCR3	Reserved. (Set to 0.)
?OCR4/ ?OCR5	Reserved. (Set to 0.)
?OCR6/ ?OCR7	Reserved. (Set to 0.)
NOTE: AOS/DVS will instead return the 128-bit unique PID of the sender in these 8 words (?OCRD through ?OCR7).	
?ORCQ	The operating system returns the request code for the function that was sent to the daemon. The codes and their meanings are the same as those in offset ?OSNQ of the ?OPSEND subpacket.
?ORSC	System call. AOS/VS returns the system call number that was issued by the requestor (?EXEC, ?OPEX, or ?OPER).
?OCR8	Reserved. (Set to 0.)
?OCR9	Waiter count. AOS/VS returns the number of outstanding requests that are queued for this daemon. If the count is zero, the daemon had to wait for a request. Otherwise, the count contains the number of outstanding send requests that were not received at the time of the call. Use this offset to return performance information.

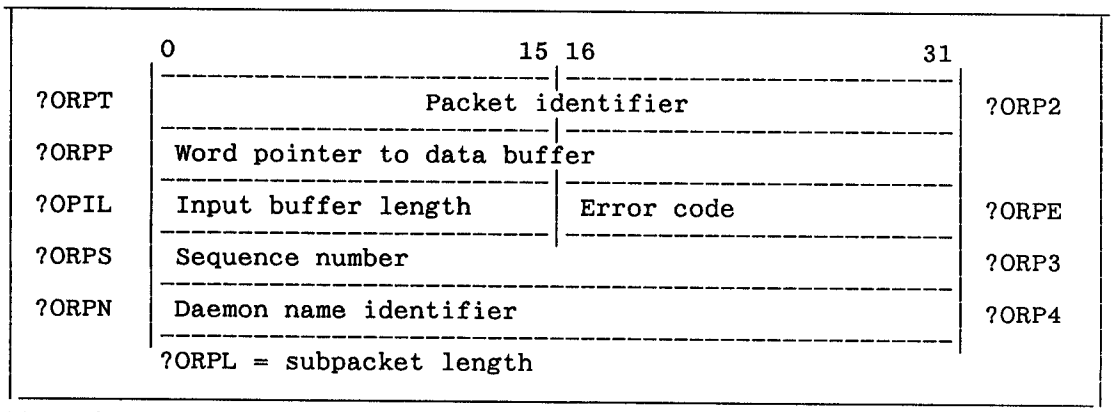


Figure 2-109. Structure of ?OPRESP Subpacket

Table 2-87. Contents of ?OPRESP Subpacket

Offset	Contents
?ORPT/ ?ORP2	Packet identifier. Place ?RSID here.
?ORPP (double word)	Data buffer pointer. This offset contains the word address of the data buffer. The buffer transfers data that is associated with particular operator requests. The daemon specifies the format of the data. Moreover, the buffer holds data that the daemon is returning to a process; it's the process that issued ?OPER/?OPSEND for this particular request.
?OPIL	Input buffer length. Supply the number of words in the input data buffer.
?ORPE	Error code. The calling daemon uses this field to notify the operating system of any errors detected during the request's processing. If no errors were encountered, then set this field to zero. Otherwise, set it to the code for the detected error.
?ORPS/ ?ORP3	Sequence number. This offset contains the unique sequence ID number that the daemon is responding to. This must be the same sequence number that the ?OPRCV function returned. This offset is required on input from the daemon, and the operating system uses this offset to map responses to the correct requestor.
?ORPN/ ?ORP4	Daemon name. Supply the name identifier of the calling daemon. If this daemon is a global type, supply the associated name identifier. If this daemon is a local one, place zero here.

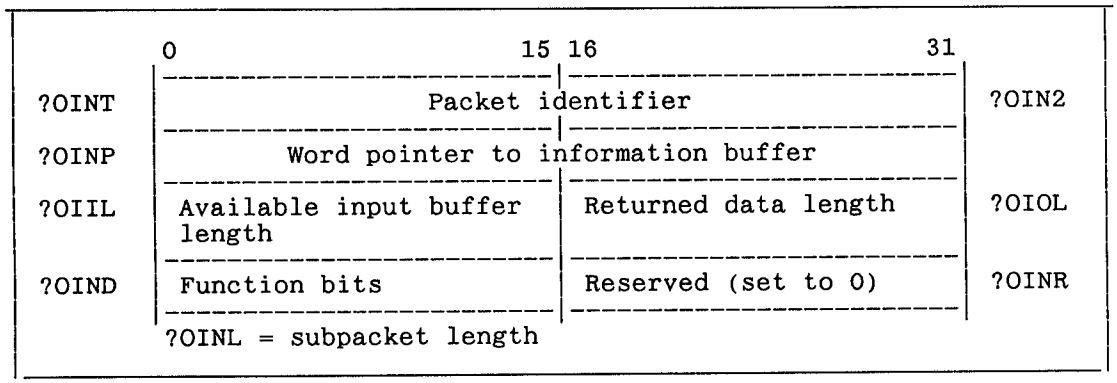


Figure 2-110. Structure of ?OPINFO Subpacket

Table 2-88. Contents of ?OPINFO Subpacket

Offset	Contents
?OINT/ ?OIN2	Packet identifier. Place ?INID here.
?OINP (double word)	Information buffer pointer. Supply the word address of a buffer that receives information from the ?OPINFO operator function. Information returned to you includes: - A bit map indicating which resources have global daemons. - A bit map indicating which of the resources have exclusively declared global daemons. - The name identifiers of daemons for a specific resource.
?OIIL	Input buffer length. This offset contains the number of words in the buffer addressed by offset ?OINP. Supply a large enough number so the buffer can hold the identifiers for all system resources.
?OIOL	Output buffer length. The operating system returns the number of words of data that it returned in the buffer.
?OIND	Function bit indicators. Supply bits to indicate the type of information you want returned to the buffer. The values of the bits and their meanings are ?OIGB--return the bit maps of system resources having a global declared daemon, and indicate which of these daemons are exclusively declared global. The operating system returns the bit maps in the first two double words of the buffer. The first double word indicates which resources have global daemons declared. The second double word indicates which of the existing global daemons are exclusively declared. The system sets corresponding bit flags. These bits are the same as in the ?OPON packet(s). The first double word will contain resource bits; the second will contain exclusive resource bits. ?OIGN--return the identifiers of global daemons that exist for the resource whose bit mask is set in offset ?ORES/ORE2 of the main packet. The identifiers are returned in the buffer that offset ?OINP points to.
?OINR	Reserved. (Set to 0.)

?OPEX Communicates between the current process and an operator process.

?OPEX *[packet address]*
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	None
Error Codes	None
Parameter Packet	Some

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?OPEX packet

Error Codes in AC0

ERBTL	Buffer too long
ERDAE	Operator already exists
ERDEB	Daemon resource error in error buffer
ERDNE	Task is not an operator
ERDRF	Daemon resource failure
ERIRB	Insufficient room in buffer
ERNOA	No operator available
ERNRQ	No matching operator request
ERPKT	Invalid packet identifier
ERPNO	Process is not an operator
ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action
ERRVN	Reserved value not zero
ERVBP	Invalid byte pointer passed as system call argument
ERVWP	Invalid word pointer passed as system call argument
ERXSP	Invalid EXEC string parameter
ERXUF	Unknown request code

Why Use It?

Use this system call to make requests of operator daemons for actions on various system resources. The actions can be service requests or status information requests.

Licensed Material - Property of Data General Corporation

?OPEX Continued

What It Does

This system call performs the functions of the CONTROL @EXEC family of commands. The call's requests end up as part of an ?OPER system call parameter packet. In turn, the packet is sent to the appropriate ?EXEC daemon for the corresponding resource request type.

You supply a main packet to any ?OPEX request. This main packet includes, in offset ?ZXFU, a function code to specify the exact request. The structure of the main packet is in Figure 2-111. Table 2-89 shows the function codes that you can place in offset ?ZXFU of the main packet. This table also explains the contents of the other offsets of the main packet.

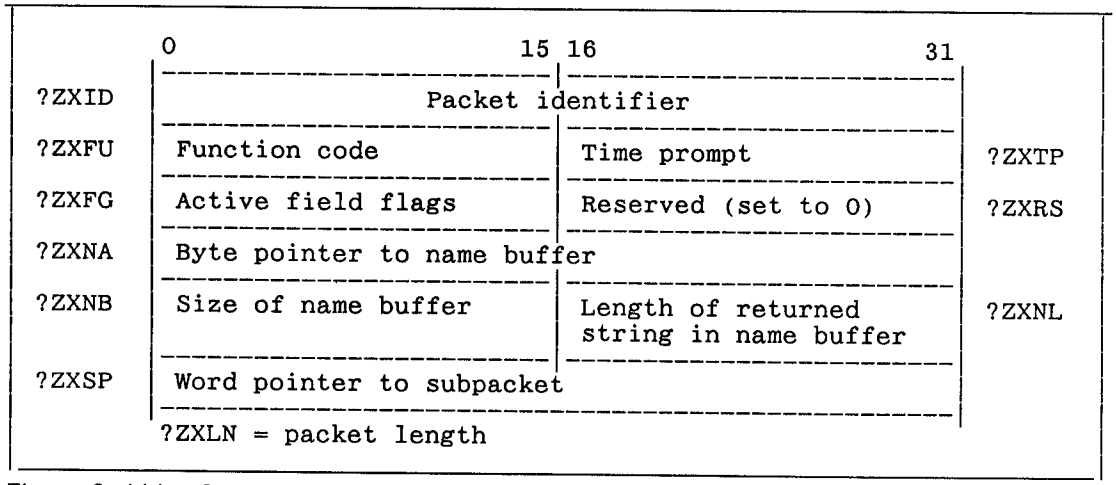


Figure 2-111. Structure of ?OPEX Main Packet

Licensed Material - Property of Data General Corporation

Table 2-89. Contents of ?OPEX Main Packet

Offset	Contents
?ZXID (double word)	Packet identifier. Place ?ZXIZ here.
?ZXFU	Function code. It specifies the ?OPEX function you want. Place one of the following values in this offset. Note that the four values with an asterisk (*) apply only to AOS/DVS.

Value	Functional Description	See Figure	See Table
?ZAC	access command	2-112	2-90
?ZAG	align command	2-113	2-91
?ZAL	allocate command		
?ZBL	batch_list command	2-114	2-92
?ZBO	batch_output command	2-115	2-93
?ZBI	binary command	2-116	2-94
?ZBR	brief command	2-117	2-95
?ZCA	cancel command	2-118	2-96
?ZCL	close command		
?ZCS	consolestatus command	2-119	2-97
?ZCO	continue command	2-120	2-98
?ZCP	cpl command	2-121	2-99
?ZCR	create command	2-122	2-100
?ZDF	defaultforms command	2-123	2-101
?ZDE	delete command		
?ZDI	disable command	2-124	2-102
?ZDS	dismounted command	2-125	2-103
?ZEL	elongate command	2-126	2-104
?ZEN	enable command	2-127	2-105
?ZEV	even command	2-128	2-106
?ZFL	flush command	2-129	2-107
?ZFN	font command		
?ZFO	forms command	2-130	2-108
?ZHA	halt command	2-131	2-109
?ZHE	headers command	2-132	2-110
?ZHO	hold command	2-133	2-111
?ZLI	limit command	2-134	2-112
?ZLO	logging command	2-135	2-113
?ZLP	lpp command	2-136	2-114
?ZMP	mapper command		
?ZME	message command		
?ZMD	modify command		
?ZMO	mounted command	2-137	2-115
?ZMS	mountstatus command	2-138	2-116
?ZON	open command		
?ZOP	operator command	2-139	2-117
?ZPA	pause command	2-140	2-118
?ZPE	premount	2-141	2-119
?ZPI	priority	2-142	2-120
?ZPR	prompts command	2-143	2-121
?ZPU	purge command		
?ZQP	queuepriority command	2-144	2-122

(continues)

?OPEX Continued

Table 2-74h. Contents of ?OPEX Main Packet

Offset	Contents			
?ZXFU (cont.)	?ZRF	refused command	2-145	2-123
	?ZRE	release command	2-146	2-124
	?ZRT	restart command	2-147	2-125
	?ZSI	silence command	2-148	2-126
	?ZSP	spoolstatus command	2-149	2-127
	?ZSK	stack command	2-150	2-128
	?ZSR	start command	2-151	2-129
	?ZSS	status command	2-152	2-130
	?ZST	stop command	2-153	2-131
	?ZTE	terminate command		
	?ZTR	trailers command	2-154	2-132
	?ZUH	unhold command	2-155	2-133
	?ZUS	unitstatus command	2-156	2-134
	?ZUL	unlimit command	2-157	2-135
	?ZUN	unsilence command	2-158	2-136
	?ZVE	verbose command	2-159	2-137
	?ZXB	xbias command	2-160	2-138
	?ZAK*	autokey command	2-161	2-139
	?ZRG*	register command	2-162	2-140
	?ZDG*	deregister command		
	?ZRS*	registerstatus command	2-163	2-141
	* These values apply only to AOS/DVS.			

(Continued)

Table 2-89. Contents of ?OPEX Main Packet

Offset	Contents
?ZXTP	Time prompt. This offset receives the time prompt from the operating system. If prompts are off, the operating system instead returns -1 to this offset.
?ZXFG	Active field flags. This offset tells the operating system that offset ?ZXNA contains a byte pointer to a buffer that you are using. For example, if the function you specify (such as ?ZAC -- the access command) requires placing a buffer byte pointer in offset ?ZXNA, then place the value ?ZYA0 (= 1B(?ZZA0)) in offset ?ZXFG. If you don't place ?ZYA0 in offset ?ZXFG, then place zero in offsets ?ZXNA and ?ZXNB. The function codes ?ZHA (halt command request), ?ZMD (modify command request), ?ZPR (prompts), ?ZRF (refused), and ?ZAK (autokey--AOS/DVS only) don't require a buffer byte pointer; place zero in offsets ?ZXNA and ?ZXNB for them.
?ZXRS	Reserved. (Set to 0.)
?ZXNA (double word)	This offset contains a byte pointer to the buffer with the device name, queue name, pathname, or username of the function you are requesting.
?ZXNB	Supply the number of bytes in the buffer that offset ?ZXNA points to.
?ZXNL	The operating system returns the number of bytes in the name that you placed in offset ?ZXNA.
?ZXSP (double word)	Supply the word address of the subpacket for the function you specified in offset ?ZXFU. If the function doesn't require a subpacket, place zero here.

(Concluded)

Licensed Material - Property of Data General Corporation

Access Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name/cooperative buffer.

Figure 2-112 shows the structure of the access command subpacket, and Table 2-90 describes its contents.

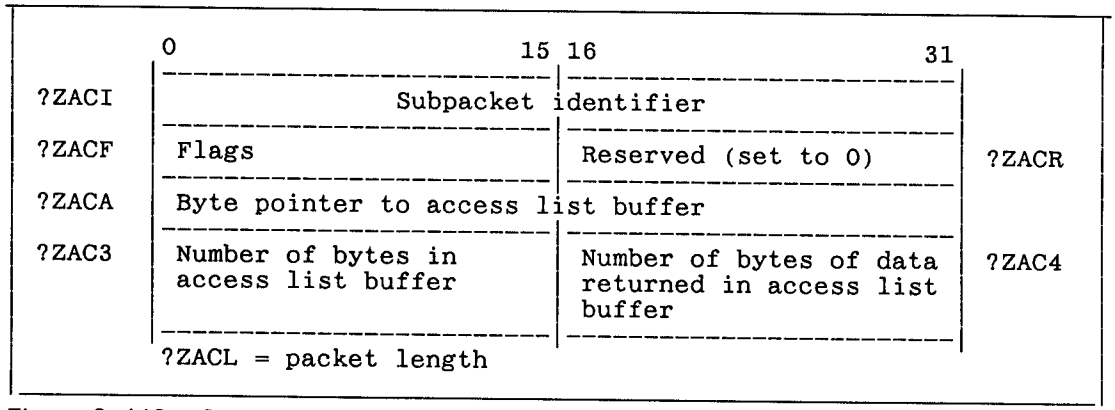


Figure 2-112. Structure of Access Command Subpacket

Table 2-90. Contents of Access Command Subpacket

Offset	Contents
?ZACI (double word)	Subpacket identifier. Place ?ZACZ here.
?ZACF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZAC. ?ZY9G = 1B(?ZZ9G) -- default. ?ZY9H = 1B(?ZZ9H) -- specify the kill option. ?ZY9I = 1B(?ZZ9I) -- operator-specified ACL.
?ZACR	Reserved. (Set to 0.)
?ZACA (double word)	Supply a byte pointer to your access list buffer.
?ZAC3	Specify the number of bytes in your access list buffer.
?ZAC4	The operating system returns the number of bytes in the access list that is in your access list buffer.

Align Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain the byte address of the device name buffer for the align command.

Figure 2–113 shows the structure of the align command subpacket, and Table 2–91 describes its contents.

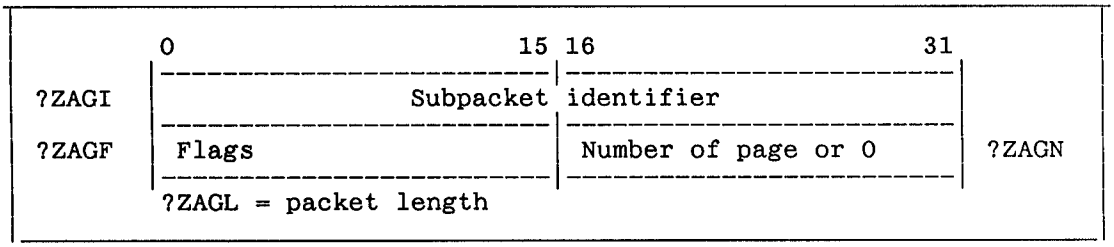


Figure 2–113. Structure of Align Command Subpacket

Table 2–91. Contents of Align Command Subpacket

Offset	Contents
?ZAGI (double word)	Subpacket identifier. Place ?ZAGZ here.
?ZAGF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZAG. ?ZYEO = 1B(?ZZEO) -- continue.
?ZAGN	Supply the page number. Or, if you supply the default page number of zero, then continuation occurs where utility program XLPT left off.

Licensed Material - Property of Data General Corporation

Allocate Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain the byte address of the device name buffer.

The allocate function does not require a subpacket.

Batch_List Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the batch queue name buffer for the batch_list function.

Figure 2-114 shows the structure of the batch_list command subpacket, and Table 2-92 describes its contents.

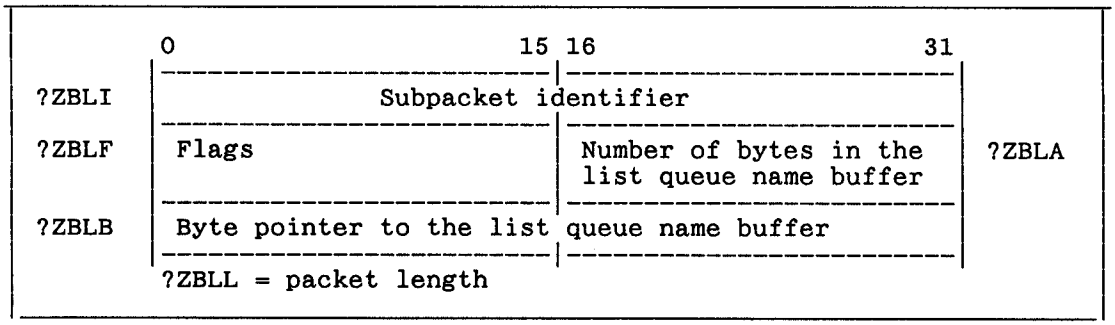


Figure 2-114. Structure of Batch_List Command Subpacket

Table 2-92. Contents of Batch_List Command Subpacket

Offset	Contents
?ZBLI (double word)	Subpacket identifier. Place ?ZBLZ here.
?ZBLF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZBL. ?ZYLO = 1B(?ZZLO) -- The default switch is on.
?ZBLA	Specify the number of bytes in the list queue name buffer.
?ZBLB (double word)	Supply a byte pointer to the list queue name buffer.

Batch_Output Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the batch queue name buffer for the batch_output function.

Figure 2-115 shows the structure of the batch_output command subpacket, and Table 2-93 describes its contents.

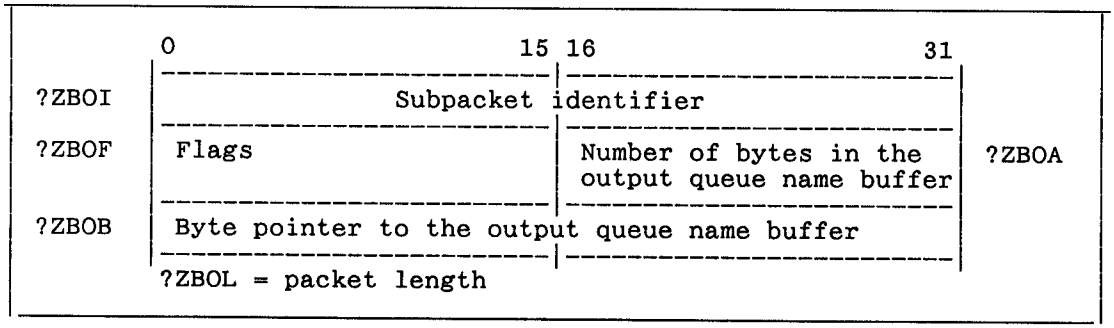


Figure 2-115. Structure of Batch_Output Command Subpacket

Table 2-93. Contents of Batch_Output Command Subpacket

Offset	Contents
?ZBOI (double word)	Subpacket identifier. Place ?ZBOZ here.
?ZBOF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZBO. ?ZYM0 = 1B(?ZZM0) -- The default switch is on.
?ZBOA	Specify the number of bytes in the output queue name buffer.
?ZBOB (double word)	Supply a byte pointer to the output queue name buffer.

Binary Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain the byte address of the device name buffer for the binary function.

Figure 2–116 shows the structure of the binary command subpacket, and Table 2–94 describes its contents.

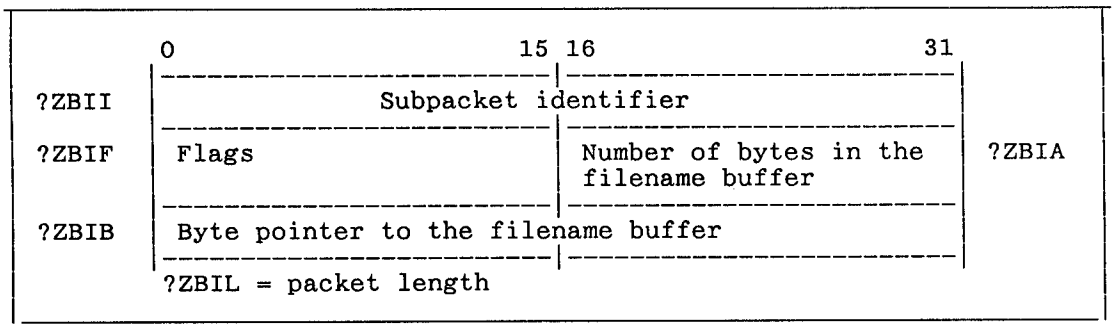


Figure 2–116. Structure of Binary Command Subpacket

Table 2–94. Contents of Binary Command Subpacket

Offset	Contents
?ZBII (double word)	Subpacket identifier. Place ?ZBIZ here.
?ZBIF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZBI. ?ZYF0 = 1B(?ZZF0) -- The byte pointer to the filename buffer is passed in offset ?ZBIB. ?ZYF1 = 1B(?ZZF1) -- Binary mode is off. ?ZJF1 = 0B(?ZZF1) -- Binary mode is on. ?ZYF2 = 1B(?ZZF2) -- Binary mode is enabled. ?ZYJ2 = 0B(?ZZF2) -- Binary mode is disabled.
?ZBIA	Specify the number of bytes in the filename buffer.
?ZBIB (double word)	Supply a byte pointer to the filename buffer. Place a filename in this buffer.

Brief Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the brief function. If you set flag bit ?ZZN0 in offset ?ZBRF of the subpacket, then the request is for all batch streams.

Figure 2-117 shows the structure of the brief command subpacket, and Table 2-95 describes its contents.

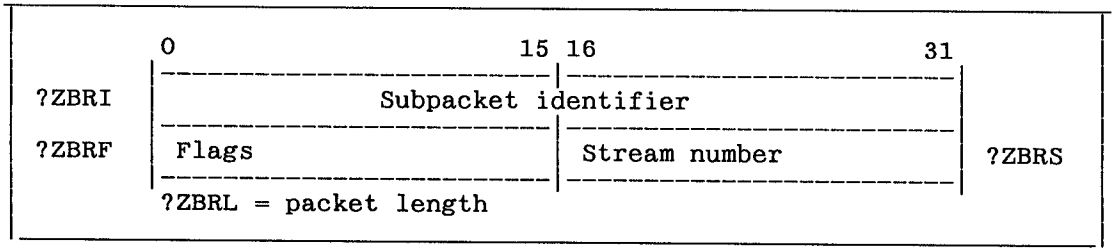


Figure 2-117. Structure of Brief Command Subpacket

Table 2-95. Contents of Brief Command Subpacket

Offset	Contents
?ZBRI (double word)	Subpacket identifier. Place ?ZBRZ here.
?ZBRF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZBR. ?ZYN0 = 1B(?ZZN0) -- All default batch input streams are specified.
?ZBRS	Supply the batch stream number if bit ?ZZN0 in offset ?ZBRF is not set. Otherwise, supply zero (for all batch streams).

Cancel Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the username buffer, if necessary, for the cancel function.

Figure 2-118 shows the structure of the cancel command subpacket, and Table 2-96 describes its contents.

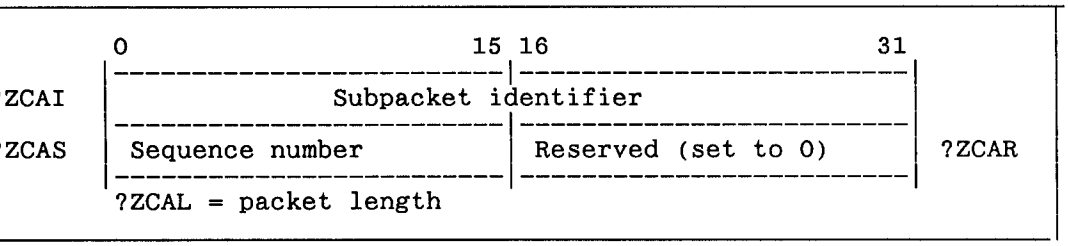


Figure 2-118. Structure of Cancel Command Subpacket

Table 2-96. Contents of Cancel Command Subpacket

Offset	Contents
?ZCAI (double word)	Subpacket identifier. Place ?ZCAZ here.
?ZCAS	Supply the sequence number in this offset.
?ZCAR	Reserved (Set to 0.)

Close Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain a byte pointer to the queue name buffer that contains the queue name for the close function.

The close function does not require a subpacket.

Consolestatus Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the consolestatus function. If you take the default (by setting bit ?ZZB0 of offset ?ZCSF), then the operating system returns the name of the console in the device name buffer.

Figure 2-119 shows the structure of the consolestatus command subpacket, and Table 2-97 describes its contents.

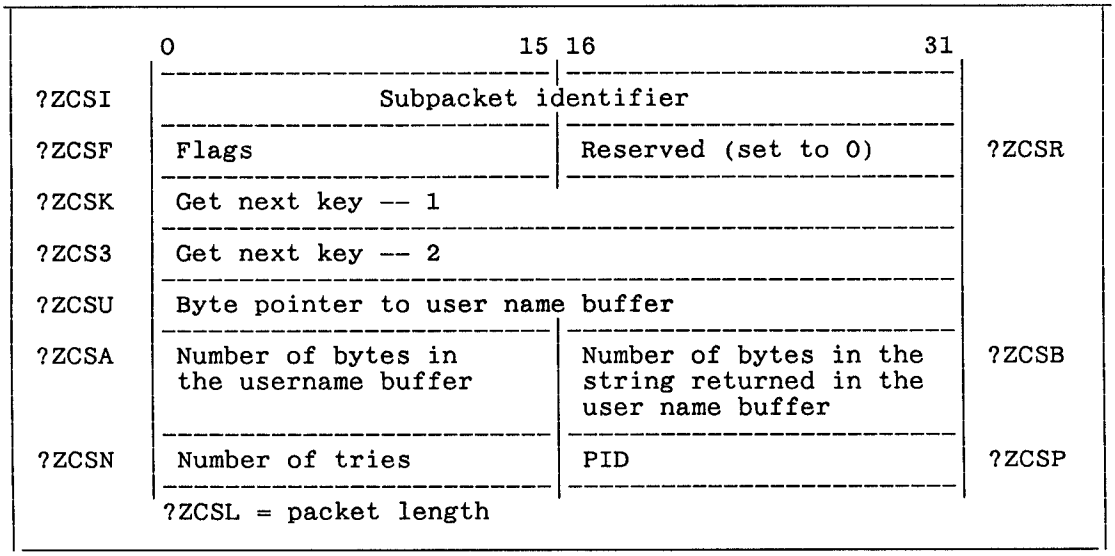


Figure 2-119. Structure of Consolestatus Command Subpacket

Table 2-97. Contents of Consolestatus Command Subpacket

Offset	Contents
?ZCSI (double word)	Subpacket identifier. Place ?ZCSZ here.
?ZCSF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZCS. ?ZYB0 = 1B(?ZZB0) -- The default. You want the status of all consoles. ?ZYB1 = 1B(?ZZB1) -- Console is enabled. ?ZYB2 = 1B(?ZZB2) -- Continue. ?ZYB3 = 1B(?ZZB3) -- Stop. ?ZYB4 = 1B(?ZZB4) -- Console is logged on. ?ZYB5 = 1B(?ZZB5) -- Console will be disabled. ?ZYB6 = 1B(?ZZB6) -- Log on/log off is in progress. ?ZYB7 = 1B(?ZZB7) -- Console is in server died state. ?ZYB8 = 1B(?ZZB8) -- Console is in error state. ?ZYB9 = 1B(?ZZB9) -- Console is not logged on.
?ZCSR	Reserved. (Set to 0.)
?ZCSK (double word)	Get next entry key. The operating system uses this offset to hold the key indicator for EXEC.
?ZCS3 (double word)	Second get next entry key. The operating system uses this offset to hold the second key indicator for EXEC.
?ZCSU (double word)	Supply a byte pointer to the buffer that receives the username.
?ZCSA	Specify a number of bytes to accommodate the largest possible username.
?ZCSB	The operating system returns the number of bytes in the username that it placed in the username buffer.
?ZCSN	The operating system returns the number of tries it made.
?ZCSP	The operating system returns the PID associated with the console.

Continue Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the continue function.

Figure 2-120 shows the structure of the continue command subpacket, and Table 2-98 describes its contents.

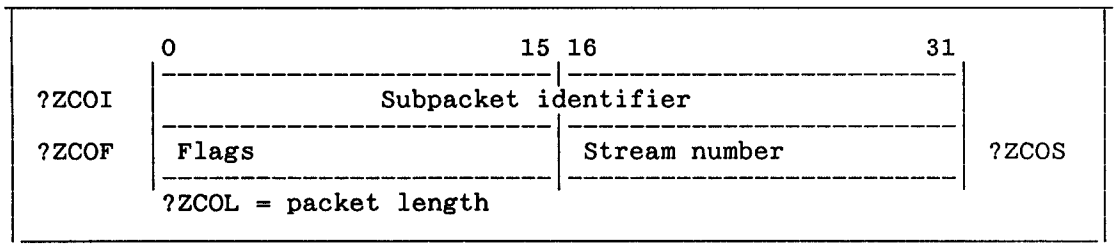


Figure 2-120. Structure of Continue Command Subpacket

Table 2-98. Contents of Continue Command Subpacket

Offset	Contents
?ZCOI (double word)	Subpacket identifier. Place ?ZCOZ here.
?ZCOF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYOO = 1B(?ZZOO) -- Continue all default batch input streams.
?ZCOS	Supply the stream number if bit ?ZZOO in offset ?ZCOF is not set. Otherwise, supply zero (for all batch input streams -- the default).

CPL (Characters Per Line) Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the cpl function.

Figure 2-121 shows the structure of the cpl command subpacket, and Table 2-99 describes its contents.

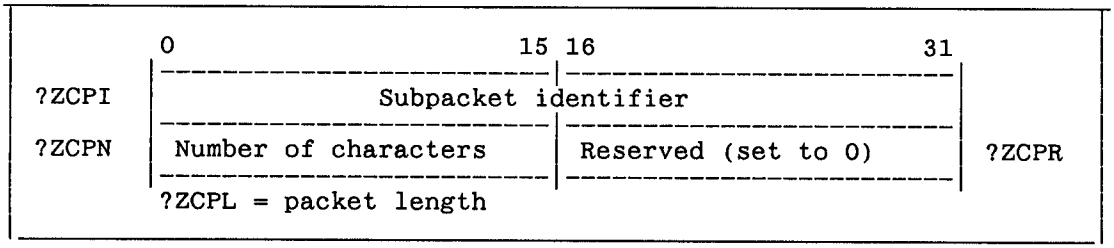


Figure 2-121. Structure of CPL Command Subpacket

Table 2-99. Contents of CPL Command Subpacket

Offset	Contents
?ZCPI (double word)	Subpacket identifier. Place ?ZCPZ here.
?ZCPN	Supply the number of characters per line.
?ZCPR	Reserved. (Set to 0.)

Create Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain the byte address of the queue name buffer for the create function.

Figure 2–122 shows the structure of the create command subpacket, and Table 2–100 describes its contents.

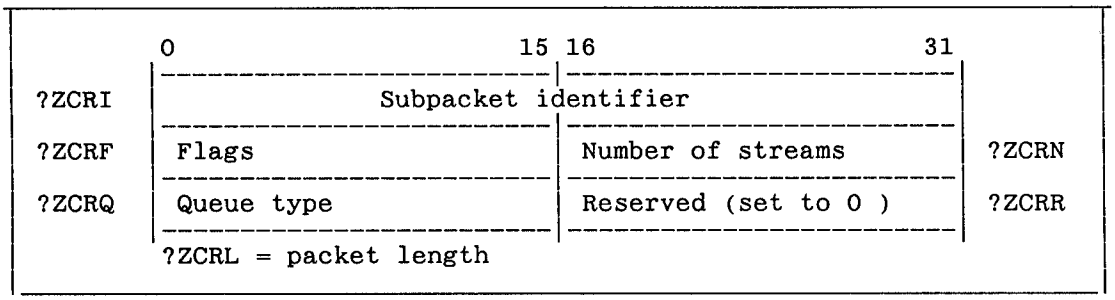


Figure 2–122. Structure of Create Command Subpacket

Table 2–100. Contents of Create Command Subpacket

Offset	Contents
?ZCRI (double word)	Subpacket identifier. Place ?ZCRZ here.
?ZCRF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZCR. ?ZYP0 = 1B(?ZZP0) --- Streams switch.
?ZCRN	Supply the number of streams.
?ZCRQ	Supply the numeric queue type.
?ZCRR	Reserved. (Set to 0.)

Defaultforms Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain the byte address of the device name buffer for the defaultforms function.

Figure 2–123 shows the structure of the defaultforms command subpacket, and Table 2–101 describes its contents.

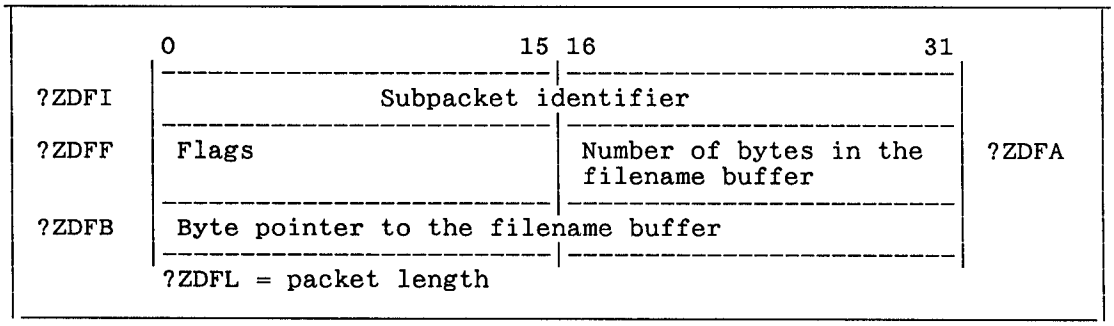


Figure 2–123. Structure of Defaultforms Command Subpacket

Table 2–101. Contents of Defaultforms Command Subpacket

Offset	Contents
?ZDFI (double word)	Subpacket identifier. Place ?ZDFZ here.
?ZDFF	Flags word. It contains the bit flags that indicate default choices to the operating system. ?ZYG0 = 1B(?ZZG0) -- If the bit is set, then the byte pointer to the filename buffer is in offset ?ZDFB. Otherwise, the default forms are used.
?ZDFA	Specify the number of bytes in the filename buffer.
?ZDFB (double word)	Supply a byte pointer to the filename buffer. Place a filename in this buffer.

Delete Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain a byte pointer to the queue name buffer that contains the queue name for the delete function.

The delete function does not require a subpacket.

Disable Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the disable function.

Figure 2-124 shows the structure of the disable command subpacket, and Table 2-102 describes its contents.

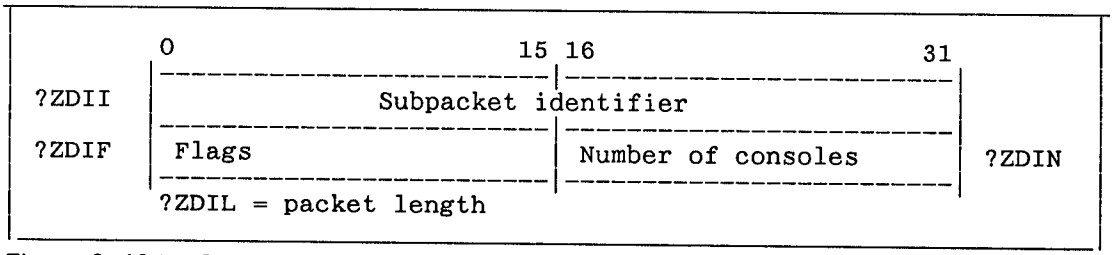


Figure 2-124. Structure of Disable Command Subpacket

Table 2-102. Contents of Disable Command Subpacket

Offset	Contents
?ZDII (double word)	Subpacket identifier. Place ?ZDIZ here.
?ZDIF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZDI. ?ZYC0 = 1B(?ZZC0) -- Default; you want to disable all the consoles. ?ZYC1 = 1B(?ZZC1) -- The console will be disabled. ?ZYC2 = 1B(?ZZC2) -- All consoles will be disabled.
?ZDIN	Number of consoles. Supply 0. If you set bit ?ZZC0 of offset ?ZDIF, AOS/VS returns the number of consoles disabled in offset ?ZDIN.

Dismounted Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the dismounted function.

Figure 2-125 shows the structure of the dismounted command subpacket, and Table 2-103 describes its contents.

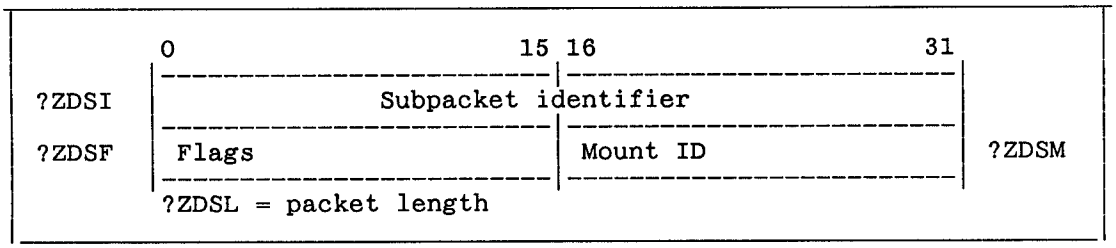


Figure 2-125. Structure of Dismounted Command Subpacket

Table 2-103. Contents of Dismounted Command Subpacket

Offset	Contents
?ZDSI (double word)	Subpacket identifier. Place ?ZDSZ here.
?ZDSF	Flags word. It contains the bit flags that indicate default choices to the operating system. ?ZY00 = 1B(?ZZ00) -- Default mount request.
?ZDSM	Supply the mount ID.

Elongate Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the elongate function.

Figure 2-126 shows the structure of the binary command subpacket, and Table 2-104 describes its contents.

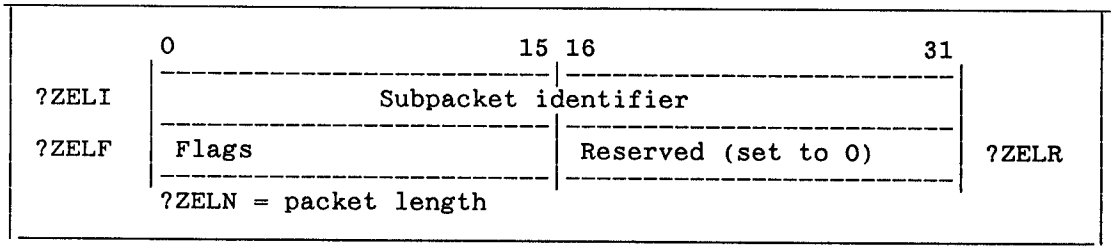


Figure 2-126. Structure of Elongate Command Subpacket

Table 2-104. Contents of Elongate Command Subpacket

Offset	Contents
?ZELI (double word)	Subpacket identifier. Place ?ZELZ here.
?ZELF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZEL. ?ZYH0 = 1B(?ZZH0) -- Elongate is turned on. ?ZJH0 = 0B(?ZZH0) -- Elongate is turned off.
?ZELR	Reserved. (Set to 0.)

Enable Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain a byte pointer to the device name buffer for the enable function. The operating system will return the console name if you specify so.

Figure 2-127 shows the structure of the enable command subpacket, and Table 2-105 describes its contents.

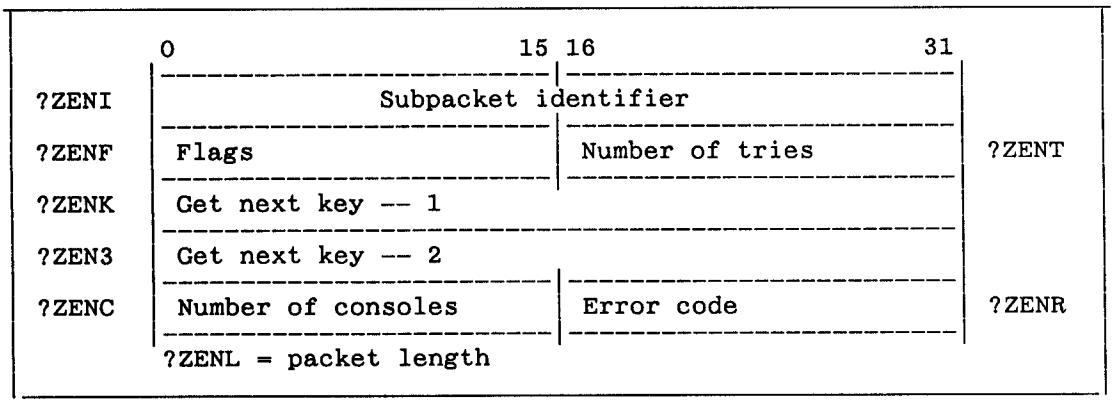


Figure 2-127. Structure of Enable Command Subpacket

Table 2-105. Contents of Enable Command Subpacket

Offset	Contents
?ZENI (double word)	Subpacket identifier. Place ?ZENZ here.
?ZENF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZEN. ?ZYD0 = 1B(?ZZD0) -- The default. You want to enable all consoles. ?ZYD1 = 1B(?ZZD1) -- Tries switch. ?ZYD2 = 1B(?ZZD2) -- Stop switch. ?ZYD3 = 1B(?ZZD3) -- Continue switch. ?ZYD4 = 1B(?ZZD4) -- Force switch. ?ZYD5 = 1B(?ZZD5) -- Console is already enabled. ?ZYD6 = 1B(?ZZD6) -- Cancelling disable, previous values are in effect. ?ZYD7 = 1B(?ZZD7) -- Cancelling disable, new log-on values are in effect. ?ZYD8 = 1B(?ZZD8) -- New log-on values are in effect. ?ZYD9 = 1B(?ZZD9) -- Console enabled. ?ZYDA = 1B(?ZZDA) -- Could not enable console. ?ZYDB = 1B(?ZZDB) -- Device already in use.
?ZENT	Supply (to EXEC) the maximum number of log-on tries.
?ZENK (double word)	Get next entry key. The operating system uses this offset to hold the key indicator for EXEC.
?ZEN3 (double word)	Second get next entry key. The operating system uses this offset to hold the second key indicator for EXEC.
?ZENC	The operating system returns the number of consoles enabled when you set the all consoles value (?ZYD0) in offset ?ZENF.
?ZENR	The operating system returns, from EXEC, an error code.

Even Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the even function.

Figure 2-128 shows the structure of the even command subpacket, and Table 2-106 describes its contents.

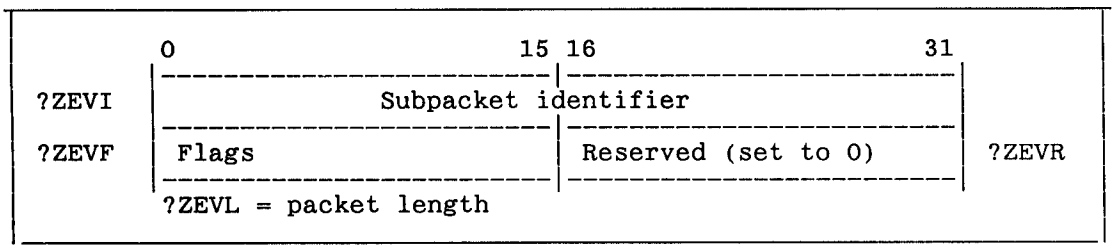


Figure 2-128. Structure of Even Command Subpacket

Table 2-106. Contents of Even Command Subpacket

Offset	Contents
?ZEVI (double word)	Subpacket identifier. Place ?ZEVZ here.
?ZEVF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for the function ?ZEV. ?ZYIO = 1B(?ZZIO) -- Even is turned on. ?ZJIO = 0B(?ZZIO) -- Even is turned off.
?ZEVR	Reserved. (Set to 0.)

?OPEX Continued

Flush Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the flush function.

Figure 2-129 shows the structure of the flush command subpacket, and Table 2-107 describes its contents.

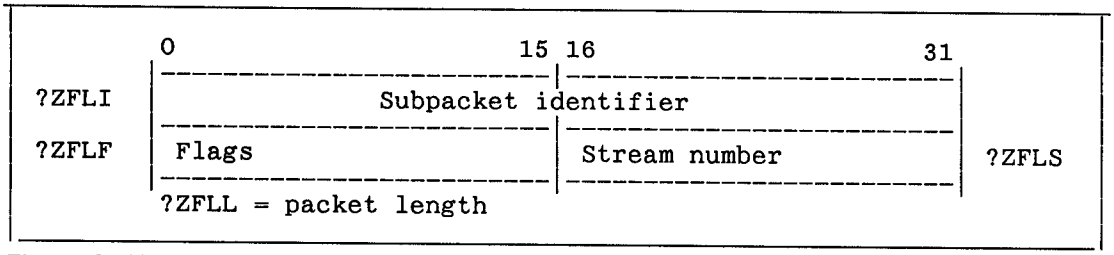


Figure 2-129. Structure of Flush Command Subpacket

Table 2-107. Contents of Flush Command Subpacket

Offset	Contents
?ZFLI (double word)	Subpacket identifier. Place ?ZFLZ here.
?ZFLF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYPG = 1B(?ZZPG) -- Flush all default batch input streams.
?ZFLS	Supply the stream number. The default is to flush all batch streams.

Font Command

This function is currently undefined.

Licensed Material - Property of Data General Corporation

Forms Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the forms function.

Figure 2-130 shows the structure of the forms command subpacket, and Table 2-108 describes its contents.

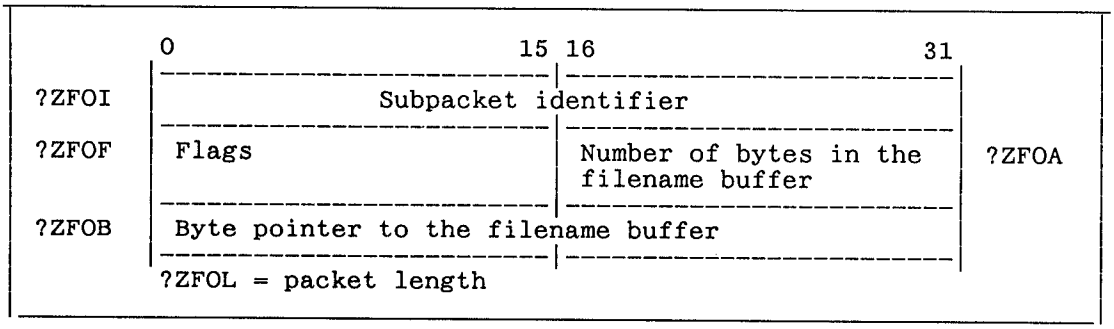


Figure 2-130. Structure of Forms Command Subpacket

Table 2-108. Contents of Forms Command Subpacket

Offset	Contents
?ZFOI (double word)	Subpacket identifier. Place ?ZFOZ here.
?ZFOF	Flags word. It contains the bit flags that indicate default choices to the operating system. ?ZYJ0 = 1B(?ZZJ0) -- If the bit is set, then the byte pointer to the filename buffer is in offset ?ZFOB. Otherwise, the implicit default forms are used.
?ZFOA	Specify the number of bytes in the filename buffer.
?ZFOB (double word)	Supply a byte pointer to the filename buffer. Place a filename in this buffer.

Licensed Material - Property of Data General Corporation

Halt Command

Bit ?ZZA0 of offset ?ZXFG of the main ?OPEX packet in Figure 2-111 must not be set, since the halt function does not require a buffer to pass information. Consequently, the operating system ignores offset ?ZXNA of the main ?OPEX packet.

Figure 2-131 shows the structure of the halt command subpacket, and Table 2-109 describes its contents.

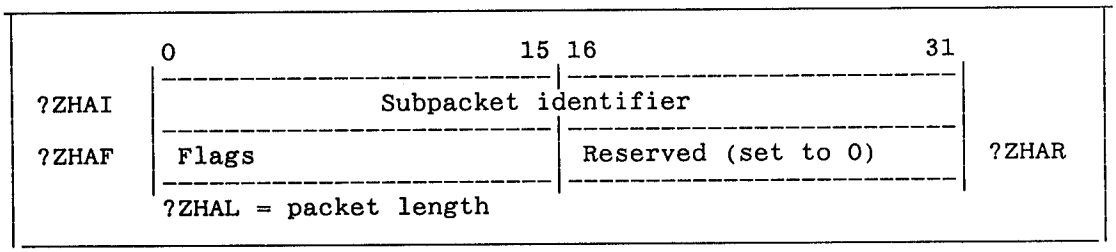


Figure 2-131. Structure of Halt Command Subpacket

Table 2-109. Contents of Halt Command Subpacket

Offset	Contents
?ZHAI (double word)	Subpacket identifier. Place ?ZHAZ here.
?ZHAF	Flags word. There are no bits defined in this word. Set it to 0.
?ZHAR	Reserved. (Set to 0.)

Headers Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the headers function.

Figure 2-132 shows the structure of the headers command subpacket, and Table 2-110 describes its contents.

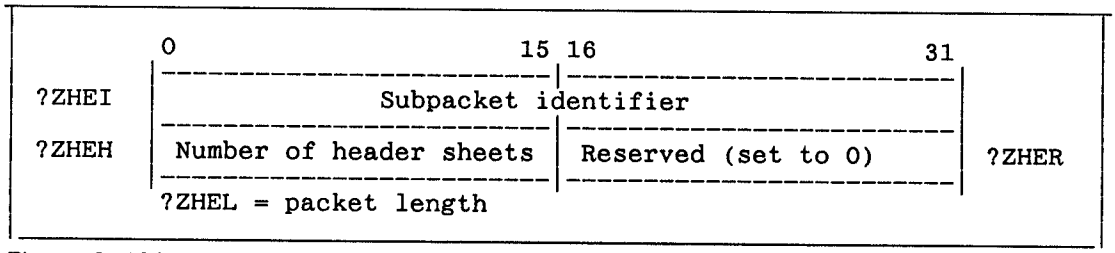


Figure 2-132. Structure of Headers Command Subpacket

Table 2-110. Contents of Headers Command Subpacket

Offset	Contents
?ZHEI (double word)	Subpacket identifier. Place ?ZHEZ here.
?ZHEH	Supply the number of header sheets before each printed job.
?ZHER	Reserved. (Set to 0.)

Hold Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the username buffer, if necessary, for the hold function.

Figure 2-133 shows the structure of the hold command subpacket, and Table 2-111 describes its contents.

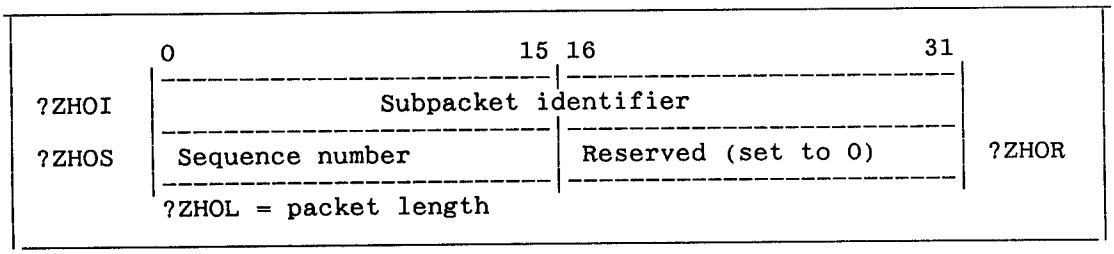


Figure 2-133. Structure of Hold Command Subpacket

Table 2-111. Contents of Hold Command Subpacket

Offset	Contents
?ZHOI (double word)	Subpacket identifier. Place ?ZHOZ here.
?ZHOS	Supply the sequence number in this offset.
?ZHOR	Reserved. (Set to 0.)

Licensed Material - Property of Data General Corporation

Limit Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the limit function.

There are two correspondences between the CLI syntax for the limit command and the ?OPEX equivalent. They are next.

```
CONTROL @EXEC LIMIT   <QUEUENAME>          [STREAM #]      [HH:MM:SS]
?OPEX         FUNC    (OPEX_PACKET.NAME)   ARGUMENT_1      ARGUMENT_2

CONTROL @EXEC LIMIT   @<DEVICENAME>        [MAX]
?OPEX         FUNC    (OPEX_PACKET.NAME)   ARGUMENT_2
```

Figure 2-134 shows the structure of the limit command subpacket, and Table 2-112 describes its contents.

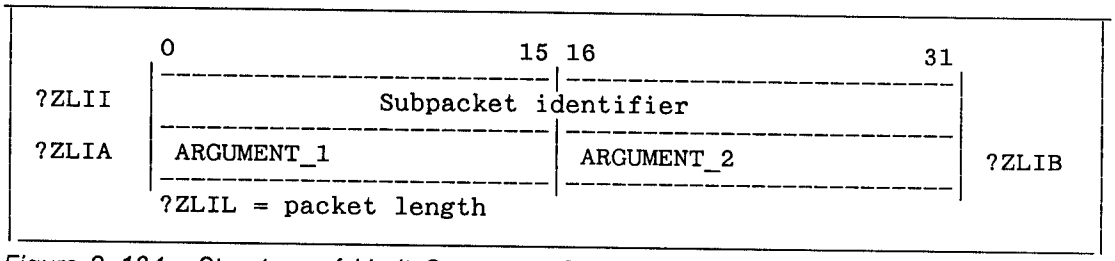


Figure 2-134. Structure of Limit Command Subpacket

Table 2-112. Contents of Limit Command Subpacket

Offset	Contents
?ZLII (double word)	Subpacket identifier. Place ?ZLIZ here.
?ZLIA	ARGUMENT_1. If you specify a queue name, supply a stream number here. The default value is all batch streams; to specify this, supply -1. If you specify a device name, then supply -1.
?ZLIB	ARGUMENT_2. If you specify a queue name, supply the time. The default time is 36:24:30; to specify this, supply -1. If you specify a device name, then supply the maximum number of pages. The default value of -1 specifies 65535 as the maximum number of pages.

Logging Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the pathname or device name buffer for the logging function. For logging status requests, this buffer will receive the current log file name from the operating system.

Figure 2-135 shows the structure of the logging command subpacket, and Table 2-113 describes its contents.

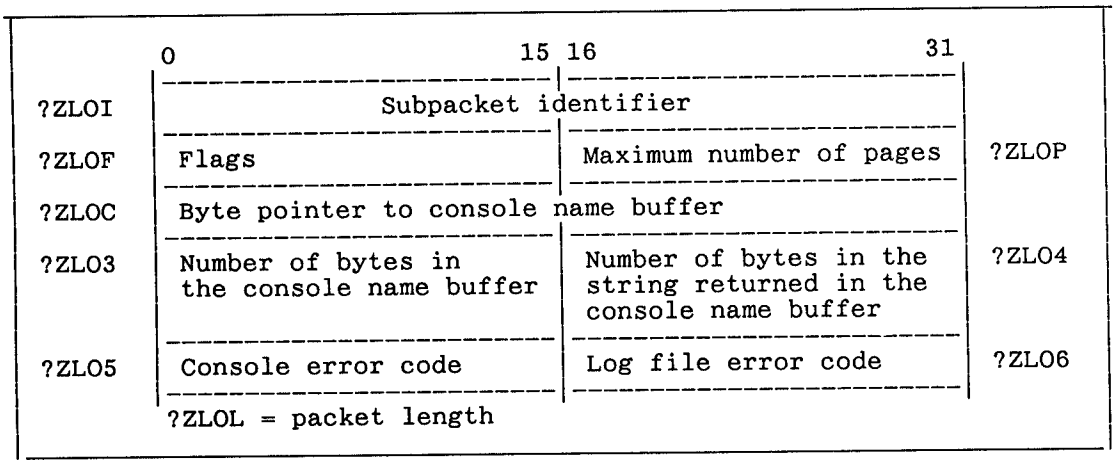


Figure 2-135. Structure of Logging Command Subpacket

Licensed Material - Property of Data General Corporation

Table 2-113. Contents of Logging Command Subpacket

Offset	Contents
?ZLOI (double word)	Subpacket identifier. Place ?ZLOZ here.
?ZLOF	Flags word. It contains the bit flags to indicate default choices to the operating system. ?ZY80 = 1B(?ZZ80) -- Return status. This is the default. ?ZY81 = 1B(?ZZ81) -- Start switch. ?ZY82 = 1B(?ZZ82) -- Maximum switch. ?ZY83 = 1B(?ZZ83) -- Stop switch. ?ZY84 = 1B(?ZZ84) -- Console switch. ?ZY85 = 1B(?ZZ85) -- No console switch. ?ZY86 = 1B(?ZZ86) -- Start logging. ?ZY87 = 1B(?ZZ87) -- Stop logging.
?ZLOP	Supply the maximum number of pages.
?ZLOC (double word)	Supply a byte pointer to the buffer that receives the console name.
?ZLO3	Specify a number of bytes to accommodate the largest possible console name.
?ZLO4	The operating system returns the number of bytes in the console name that it placed in the console name buffer.
?ZLO5	EXEC returns a console error code to this offset.
?ZLO6	EXEC returns a log file error code to this offset.

LPP (Lines Per Page) Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2–111 must contain the byte address of the device name buffer for the lpp function.

Figure 2–136 shows the structure of the lpp command subpacket, and Table 2–114 describes its contents.

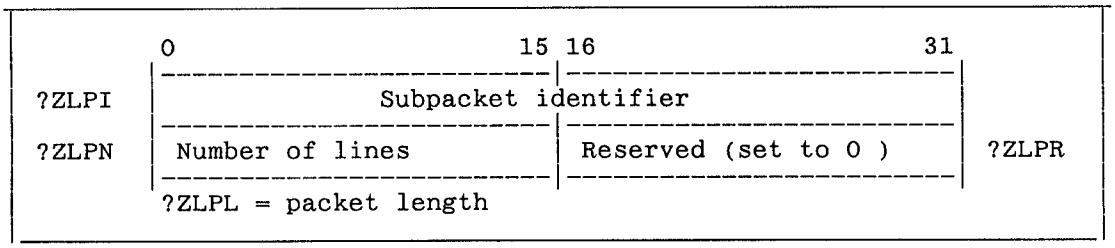


Figure 2–136. Structure of LPP Command Subpacket

Table 2–114. Contents of LPP Command Subpacket

Offset	Contents
?ZLPI (double word)	Subpacket identifier. Place ?ZLPZ here.
?ZLPN	Supply the number of lines per page.
?ZLPR	Reserved. (Set to 0.)

?OPEX Continued

Mapper Command

This function is currently undefined.

Licensed Material - Property of Data General Corporation

Message Command

This function does not require a subpacket. Instead, supply a byte pointer to the message text buffer in offset ?ZXNA of the ?OPEX main packet in Figure 2–111.

Modify Command

Supply 0 in offset ?ZXNA of the ?OPEX main packet in Figure 2-111, since all information is passed in the subpacket. The subpacket has the same format as the one for the ?XFMOD function of system call ?EXEC. See Figure 2-36.

Mounted Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer, if necessary, for the mounted function.

Figure 2-137 shows the structure of the mounted command subpacket, and Table 2-115 describes its contents.

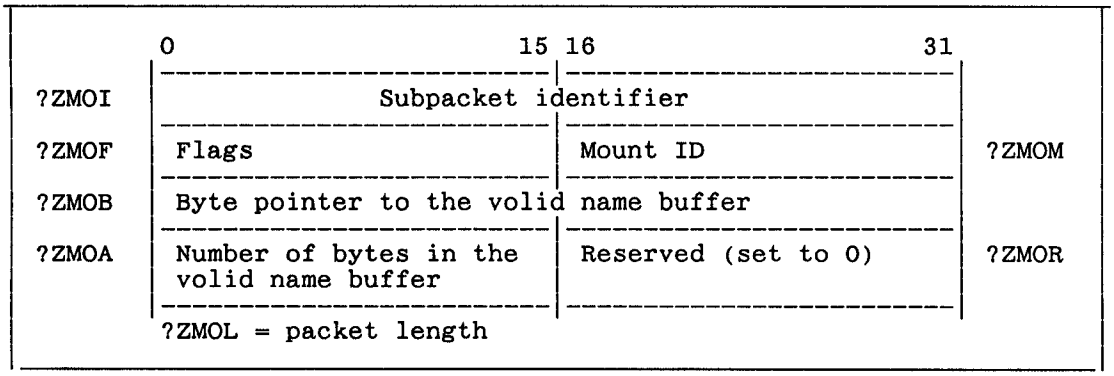


Figure 2-137. Structure of Mounted Command Subpacket

Table 2-115. Contents of Mounted Command Subpacket

Offset	Contents
?ZMOI (double word)	Subpacket identifier. Place ?ZMOZ here.
?ZMOF	Flags word. It contains the bit flags that indicate default choices to the operating system. ?ZY10 = 1B(?ZZ10) -- Default mount ID.
?ZMOM	Specify the mount ID.
?ZMOB (double word)	Supply a byte pointer to the volid name buffer.
?ZMOA	Specify the number of bytes in the volid name buffer.
?ZMOR	Reserved. (Set to 0.)

Mountstatus Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 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 2-138 shows the structure of the mountstatus command subpacket, and Table 2-116 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 volid name buffer		
?ZMS7	Number of bytes in the volid name buffer	Number of bytes in the string returned in the volid 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 volid list name buffer		
?ZMSA	Number of bytes in the volid list name buffer	Number of bytes in the string returned in the volid list name buffer	?ZMSZ
?ZMSM	Mount ID	Mount error	?ZMSR

?ZMSL = packet length

Figure 2-138. Structure of Mountstatus Command Subpacket

Table 2-116. 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. ?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 volid 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.

(Continues)

Table 2-116. Contents of Mountstatus Command Subpacket

Offset	Contents
?ZMSX (double word)	Second get next entry key. The operating system uses this offset to hold the second key indicator for EXEC.
?ZMSU (double word)	Supply a byte pointer to the buffer that receives the username.
?ZMS3	Specify a number of bytes to accommodate the largest possible username.
?ZMS4	The operating system returns the number of bytes in the username that it placed in the username buffer.
?ZMST (double word)	Supply a byte pointer to the buffer that receives the request text.
?ZMS5	Specify a number of bytes to accommodate the largest possible request text.
?ZMS6	The operating system returns the number of bytes in the text that it placed in the request text buffer.
?ZMSV (double word)	Supply a byte pointer to the buffer that receives the volid.
?ZMS7	Specify a number of bytes to accommodate the largest possible volid.
?ZMS8	The operating system returns the number of bytes in the volid that it placed in the volid buffer.
?ZMSD (double word)	Supply a byte pointer to the buffer that receives the device name. The content is a list of device names that are separated by nulls; a pair of nulls terminates the list.
?ZMS9	Specify a number of bytes to accommodate the largest possible device name. The current maximum size of a device name list is 1024. bytes.
?ZMS0	The operating system returns the number of bytes in the data that it placed in the device list buffer.
?ZMSS (double word)	Supply a byte pointer to the buffer that receives the volid list.
?ZMSA	Specify a number of bytes to accommodate the largest possible volid list.
?ZMSZ	The operating system returns the number of bytes in the volid list that it placed in the volid list buffer.
?ZMSM	Supply the mount ID.
?ZMSR	The operating system returns the mount error code.

(Concluded)

Open Command

This function does not require a subpacket. Instead, supply a byte pointer to the queue name buffer that contains the queue name for the open function. This byte pointer belongs in offset ?ZXNA of the ?OPEX main packet in Figure 2–111.

Operator Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the operator function.

Figure 2-139 shows the structure of the operator command subpacket, and Table 2-117 describes its contents.

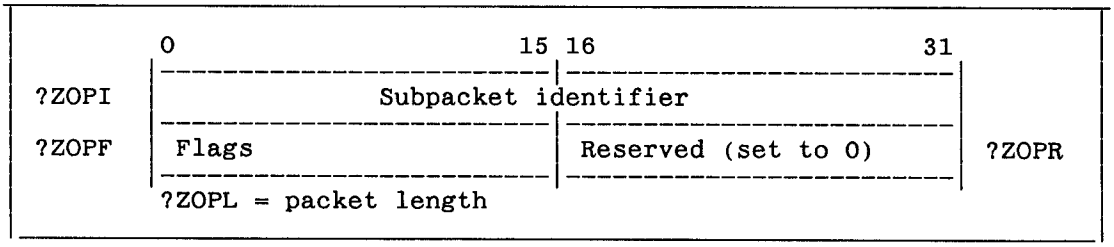


Figure 2-139. Structure of Operator Command Subpacket

Table 2-117. Contents of Operator Command Subpacket

Offset	Contents
?ZOPI (double word)	Subpacket identifier. Place ?ZOPZ here.
?ZOPF	Flags word. It contains bit flags with additional information to the operating system and EXEC for function ?ZOP. ?ZJ30 = 0B(?ZZ30) -- There is no operator. ?ZY30 = 1B(?ZZ30) -- There is an operator on duty.
?ZOPR	Reserved. (Set to 0.)

Pause Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the pause function.

Figure 2-140 shows the structure of the pause command subpacket, and Table 2-118 describes its contents.

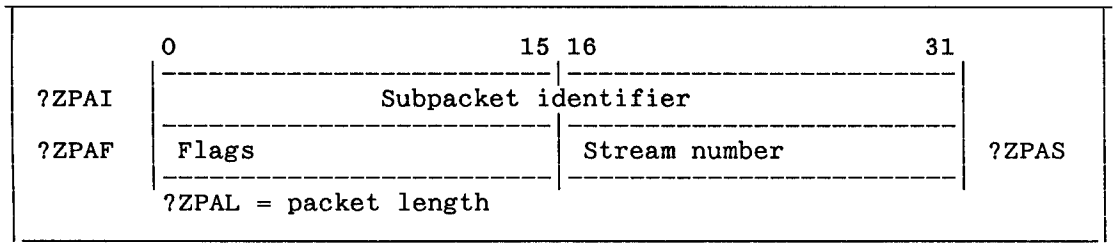


Figure 2-140. Structure of Pause Command Subpacket

Table 2-118. Contents of Pause Command Subpacket

Offset	Contents
?ZPAI (double word)	Subpacket identifier. Place ?ZPAZ here.
?ZPAF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYQ0 = 1B(?ZZQ0) -- Pause all default batch input streams.
?ZPAS	Supply the stream number. The default (the value 0) is to pause all batch streams.

Premount Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the premount function.

Figure 2-141 shows the structure of the premount command subpacket, and Table 2-119 describes its contents.

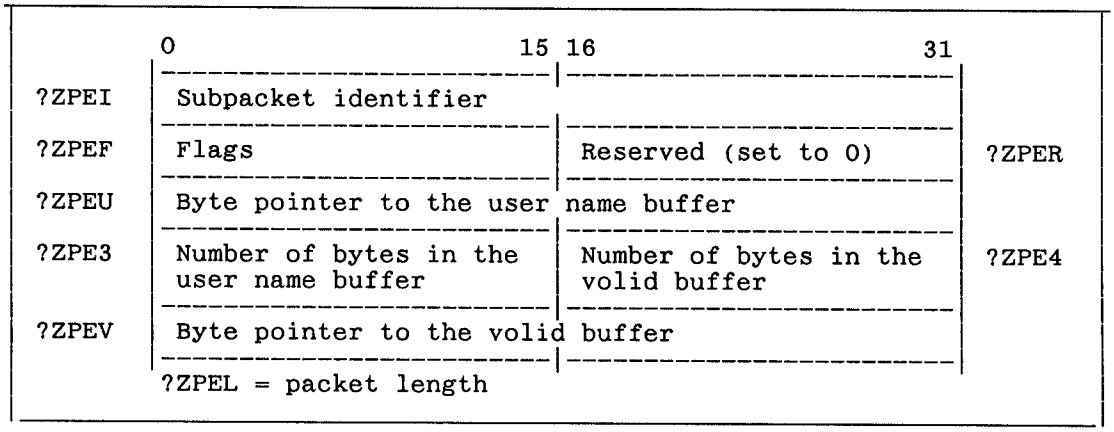


Figure 2-141. Structure of Premount Command Subpacket

Table 2-119. Contents of Premount Command Subpacket

Offset	Contents
?ZPEI (double word)	Subpacket identifier. Place ?ZPEZ here.
?ZPEF	Flags word. It contains bit flags with additional information to the operating system and EXEC for function ?ZPE. ?ZY40 = 1B(?ZZ40) -- IBM switch.
?ZPER	Reserved (Set to 0.)
?ZPEU (double word)	Supply a byte pointer to the buffer that receives the username.
?ZPE3	Specify the number of bytes in the username buffer.
?ZPE4	Specify the number of bytes in the volid buffer.
?ZPEV (double word)	Supply a byte pointer to the buffer that receives the volid.

Licensed Material - Property of Data General Corporation

Priority Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue or cooperative simple process name buffer for the priority function.

There are two correspondences between the CLI syntax for the priority command and the ?OPEX equivalent. They are next.

```
CONTROL @EXEC PRIORITY[/SWITCH] <QUEUENAME> [STREAM #]   [PRIORITY]
?OPEX         FUNC     (OPEX_PACKET.NAME)     ARGUMENT_1  ARGUMENT_2

CONTROL @EXEC PRIORITY[/SWITCH] @<DEVICENAME> [PRIORITY]
?OPEX         FUNC     (OPEX_PACKET.NAME)       ARGUMENT_2
```

Figure 2-142 shows the structure of the priority command subpacket, and Table 2-120 describes its contents.

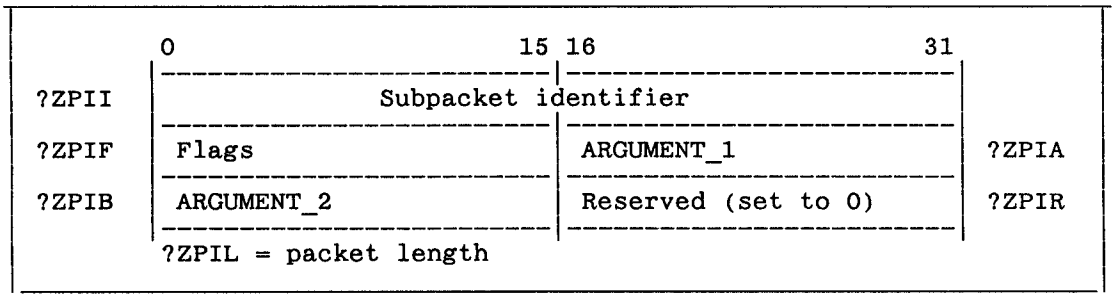


Figure 2-142. Structure of Priority Command Subpacket

Table 2-120. Contents of Priority Command Subpacket

Offset	Contents
?ZPII (double word)	Subpacket identifier. Place ?ZPIZ here.
?ZPIF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZPI. ?ZYR0 = 1B(?ZZR0) -- Process type is swappable. ?ZYR1 = 1B(?ZZR1) -- Process type is pre-emptible. ?ZYR2 = 1B(?ZZR2) -- Process type is resident.
?ZPIA	ARGUMENT_1. If you specify a queue name, supply a stream number here. The default value is all batch streams; to specify this, supply -1. If you specify a device name, then the operating system ignores this offset. (Set it to 0.)
?ZPIB	ARGUMENT_2. If you specify a queue name, supply its priority. If you specify a device name, supply its priority.
?ZPIR	Reserved. (Set to 0.)

Prompts Command

Bit ?ZZA0 of offset ?ZXFG of the main ?OPEX packet in Figure 2-111 must not be set, since the prompts function does not require a buffer to pass information. Consequently, the operating system ignores offset ?ZXNA of the main ?OPEX packet.

Figure 2-143 shows the structure of the prompts command subpacket, and Table 2-121 describes its contents.

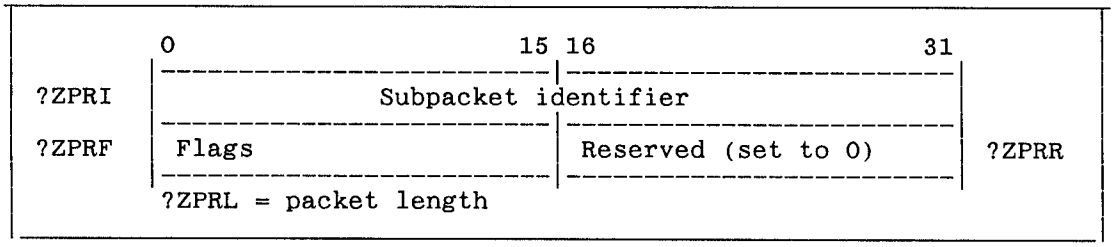


Figure 2-143. Structure of Prompts Command Subpacket

Table 2-121. Contents of Prompts Command Subpacket

Offset	Contents
?ZPRI (double word)	Subpacket identifier. Place ?ZPRZ here.
?ZPRF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZPR. ?ZJ90 = 0B(?ZZ90) -- Prompts off. ?ZY90 = 1B(?ZZ90) -- Prompts on.
?ZPRR	Reserved. (Set to 0.)

Purge Command

This function does not require a subpacket. Instead, supply a byte pointer to the queue name buffer. This buffer contains the queue name for the purge function. Supply the byte pointer in offset ?ZXNA of the ?OPEX main packet in Figure 2–111.

Qpriority Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the qpriority function.

There are two correspondences between the CLI syntax for the qpriority command and the ?OPEX equivalent. They are next.

```
CONTROL @EXEC QPRIORITY [QUEUENAME]          [STREAM #] [HI PRIO]  [LO PRIO]
?OPEX          FUNC      (OPEX_PACKET.NAME) ARG_1      ARG_2      ARG_3

CONTROL @EXEC QPRIORITY @DEVICENAME          [HIGH PRIORITY] [LOW PRIORITY]
?OPEX          FUNC     (OPEX_PACKET.NAME)  ARG_2           ARG_3
```

Figure 2-144 shows the structure of the qpriority command subpacket, and Table 2-122 describes its contents.

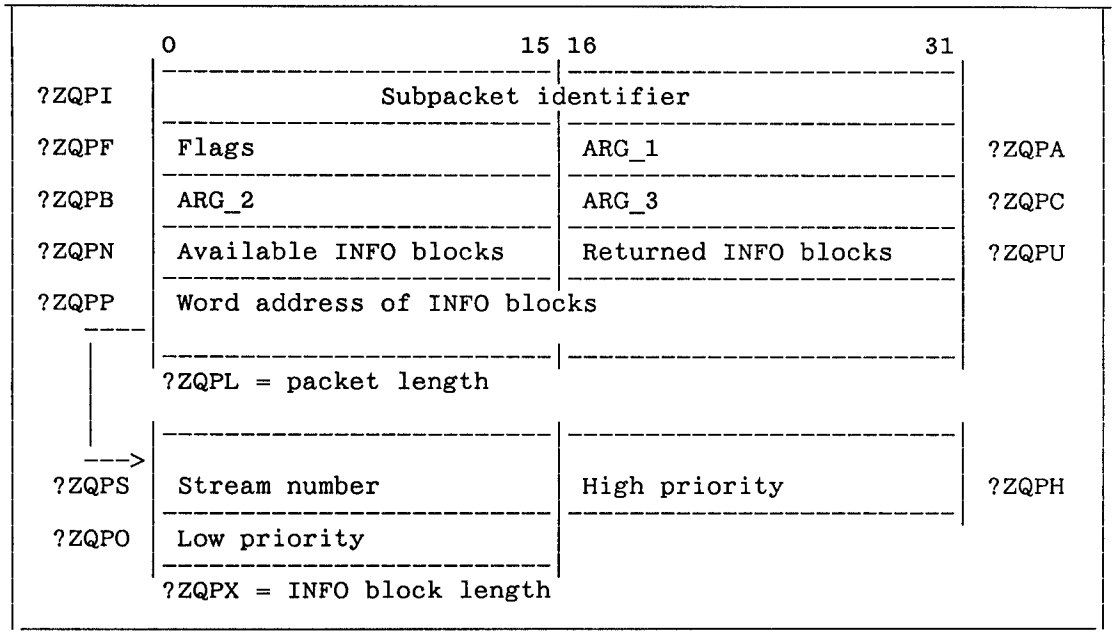


Figure 2-144. Structure of Qpriority Command Subpacket

Licensed Material - Property of Data General Corporation

Table 2-122. Contents of Qpriority Command Subpacket

Offset	Contents
?ZQPI (double word)	Subpacket identifier. Place ?ZQPZ here.
?ZQPF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYS0 = 1B(?ZZS0) -- All default batch input streams. ?ZJS1 = 0B(?ZZS1) -- Display information. ?ZYS1 = 1B(?ZZS1) -- Set information.
?ZQPA	ARG_1. If you specify a queue name, supply a stream number. If you specify a device name, then the operating system ignores this offset. (Set it to 0.)
?ZQPB	ARG_2. If you specify a queue name, supply the high-priority value. If you specify a device name and ?ZYS1 in offset ?ZQPF, supply the high-priority value. If you specify a device name and ?ZJS1 in offset ?ZQPF, supply zero.
?ZQPC	ARG_3. If you specify a queue name, supply the low-priority value. If you specify a device name and ?ZYS1 in offset ?ZQPF, supply the low-priority value. If you specify a device name and ?ZJS1 in offset ?ZQPF, supply zero.
?ZQPN	Specify the number of available INFO blocks, but only if offset ?ZQPF contains ?ZJS1 (to display information). Otherwise, supply 0.
?ZQPU	The operating system returns the actual number of INFO blocks, but only if offset ?ZQPF contains ?ZJS1 (to display information. Otherwise, supply 0.
?ZQPP	Supply the word address of the first of the INFO blocks, but only if offset ?ZQPF contains ?ZJS1 (to display information). Otherwise, supply 0.
?ZQPS	The operating system returns the INFO block stream number.
?ZQPH	The operating system returns the INFO block high-priority value.
?ZQPO	The operating system returns the INFO block low-priority value.

Refused Command

Bit ?ZZA0 of offset ?ZXFG of the main ?OPEX packet in Figure 2-111 must not be set, since the refused function does not require a buffer to pass information. Consequently, the operating system ignores offset ?ZXNA of the main ?OPEX packet.

Figure 2-145 shows the structure of the refused command subpacket, and Table 2-123 describes its contents.

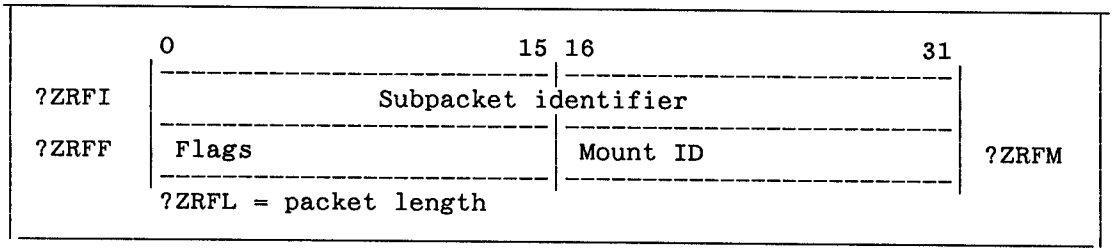


Figure 2-145. Structure of Refused Command Subpacket

Table 2-123. Contents of Refused Command Subpacket

Offset	Contents
?ZRFI (double word)	Subpacket identifier. Place ?ZRFZ here.
?ZRFF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZY50 = 1B(?ZZ50) -- Default mount request.
?ZRFM	Supply the mount ID. If you specify the default in offset ?ZRFF, then it applies to the default mount request.

Release Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the release function.

Figure 2-146 shows the structure of the release command subpacket, and Table 2-124 describes its contents.

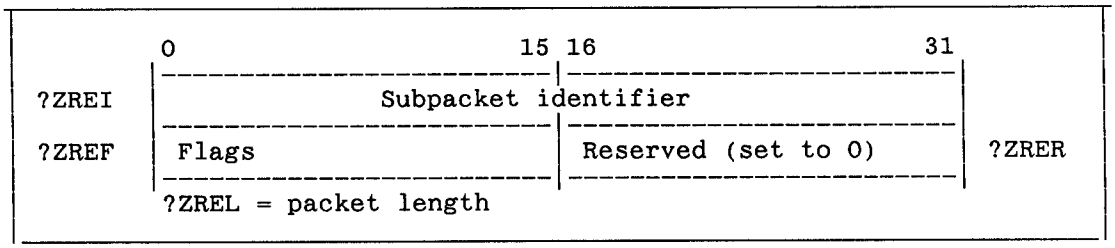


Figure 2-146. Structure of Release Command Subpacket

Table 2-124. Contents of Release Command Subpacket

Offset	Contents
?ZREI (double word)	Subpacket identifier. Place ?ZREZ here.
?ZREF	Flags word. It contains bit flags with additional information to the operating system and EXEC for function ?ZRE. ?ZY60 = 1B(?ZZ60) -- All switch.
?ZRER	Reserved. (Set to 0.)

Restart Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the restart function.

There are two correspondences between the CLI syntax for the restart command and the ?OPEX equivalent. They are next.

```
CONTROL @EXEC RESTART [QUEUENAME]           [STREAM #]
?OPEX         FUNC    (OPEX_PACKET.NAME)    ARGUMENT_1

CONTROL @EXEC RESTART @DEVICENAME           [BEG. PAGE #]  [END PAGE #]
?OPEX         FUNC    (OPEX_PACKET.NAME)    ARGUMENT_1     [ARGUMENT_2]
```

Figure 2-147 shows the structure of the restart command subpacket, and Table 2-125 describes its contents.

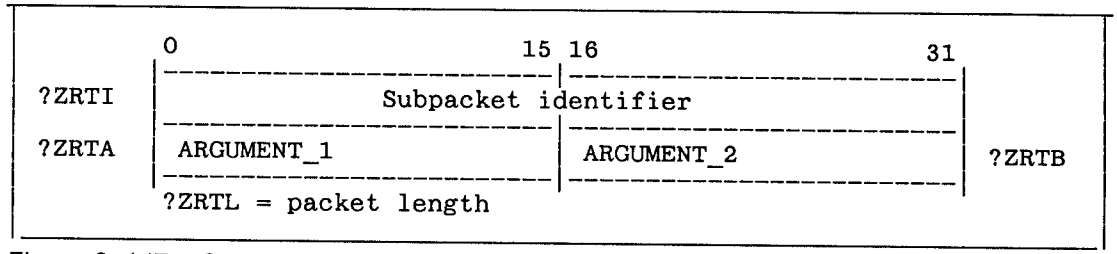


Figure 2-147. Structure of Restart Command Subpacket

Table 2-125. Contents of Restart Command Subpacket

Offset	Contents
?ZRTI (double word)	Subpacket identifier. Place ?ZRTZ here.
?ZRTA	ARGUMENT_1. If you specify a queue name, supply a stream number here. If you specify a device name, then supply the beginning page number. The default is the first page; to specify this, supply -1.
?ZRTB	ARGUMENT_2. If you specify a queue name, the operating system ignores this offset. (Set it to 0.) If you specify a device name, then supply the ending page number. The default is the last page; to specify this, supply -1.

Silence Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the silence function.

Figure 2-148 shows the structure of the silence command subpacket, and Table 2-126 describes its contents.

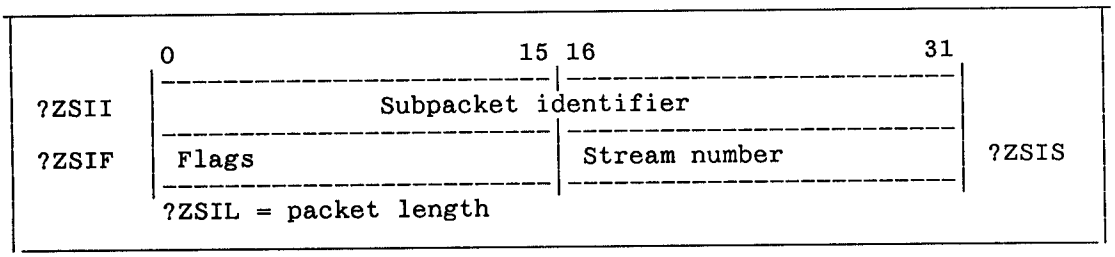


Figure 2-148. Structure of Silence Command Subpacket

Table 2-126. Contents of Silence Command Subpacket

Offset	Contents
?ZSII (double word)	Subpacket identifier. Place ?ZSIZ here.
?ZSIF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYTO = 1B(?ZZTO) -- Silence all batch input streams.
?ZSIS	Supply the number of the stream if bit ?ZZTO in offset ?ZSIF is off. Otherwise, supply 0.

Spoolstatus Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the spoolstatus function. To obtain the status of all spooled devices, set bit ?ZZK0 of offset ?ZSPF of the subpacket.

Figure 2-149 shows the structure of the spoolstatus command subpacket, and Table 2-127 describes its contents.

	0 15	16 31	
?ZSPI	Subpacket identifier		
?ZSPF	Flags	Limit value	?ZSPV
?ZSPC	Characters per line	Lines per page	?ZSPN
?ZSPH	Number of headers	Number of trailers	?ZSPT
?ZSPB	Bias factor	Priority	?ZSPP
?ZSPK	Get next key -- 1		
?ZSP9	Get next key -- 2		
?ZSPQ	Byte pointer to queue name list buffer		
?ZSP3	Number of bytes in the queue name list buffer	Number of bytes in the string returned in the queue name list buffer	?ZSP4
?ZSPD	Byte pointer to default forms name buffer		
?ZSP5	Number of bytes in the default forms name buffer	Number of bytes in the string returned in the default forms name buffer	?ZSP6
?ZSPS	Byte pointer to special forms name buffer		
?ZSP7	Number of bytes in the special forms name buffer	Number of bytes in the string returned in the special forms name buffer	?ZSP8
?ZSPE	Error code	Reserved (set to 0)	?ZSPR

?ZSPL = packet length

Figure 2-149. Structure of Spoolstatus Command Subpacket

Table 2-127. Contents of Spoolstatus Command Subpacket

Offset	Contents
?ZSPI (double word)	Subpacket identifier. Place ?ZSPZ here.
?ZSPF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYK0 = 1B(?ZZK0) -- You want the spool status of all devices. ?ZYK1 = 1B(?ZZK1) -- An output bit; even mode is enabled. ?ZYK2 = 1B(?ZZK2) -- An output bit; binary mode is enabled. ?ZYK3 = 1B(?ZZK3) -- An output bit; uppercase is enabled. ?ZYK4 = 1B(?ZZK4) -- An output bit; new line conversion is enabled. ?ZYK5 = 1B(?ZZK5) -- An output bit; process type is swappable. ?ZYK6 = 1B(?ZZK6) -- An output bit; process type is pre-emptible. ?ZYK7 = 1B(?ZZK7) -- An output bit; process type is resident. ?ZYK8 = 1B(?ZZK8) -- An output bit; default forms are specified. ?ZYK9 = 1B(?ZZK9) -- An output bit; special forms are specified. ?ZYKA = 1B(?ZZKA) -- An output bit; limiting is enabled (the default value). ?ZYKB = 1B(?ZZKB) -- An output bit; print or plot device.
?ZSPV	The operating system returns the limit value.
?ZSPC	The operating system returns the number of characters per line.
?ZSPN	The operating system returns the number of lines per page.
?ZSPH	The operating system returns the number of headers.
?ZSPT	The operating system returns the number of trailers.
?ZSPB	The operating system returns the bias factor.
?ZSPP	The operating system returns the priority.

(Continues)

Table 2-127. Contents of Spoolstatus Command Subpacket

Offset	Contents
?ZSPK (double word)	Get next key entry. The operating system uses this offset to hold the next key entry for EXEC.
?ZSP9 (double word)	Second get next key entry. The operating system uses this offset to hold the second next key entry for EXEC.
?ZSPQ (double word)	The operating system returns a byte pointer to the queue name buffer. In this buffer, nulls separate the queue names and a pair of nulls terminates the queue names.
?ZSP3	Specify the number of bytes in the queue name buffer. The largest possible queue name list is 1024. bytes.
?ZSP4	The operating system returns the number of bytes in the queue name list that it placed in the queue name buffer.
?ZSPD (double word)	Supply a byte pointer to the buffer that receives the default forms name.
?ZSP5	Specify a number of bytes to accommodate the largest possible default forms name.
?ZSP6	The operating system returns the number of bytes in the default forms filename that it placed in the default forms name buffer.
?ZSPS (double word)	Supply a byte pointer to the buffer that receives the special forms name.
?ZSP7	Specify a number of bytes to accommodate the largest possible special forms name.
?ZSP8	The operating system returns the number of bytes in the special forms filename that it placed in the special forms name buffer.
?ZSPE	The operating system returns an error code that EXEC gave to it.
?ZSPR	Reserved. (Set to 0.)

(Concluded)

Stack Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of a pathname buffer. This buffer contains the pathname for the stack function.

Figure 2-150 shows the structure of the stack command subpacket, and Table 2-128 describes its contents.

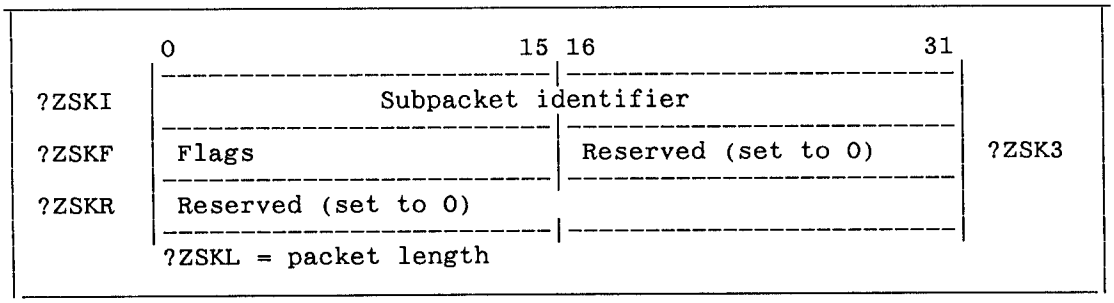


Figure 2-150. Structure of Stack Command Subpacket

Table 2-128. Contents of Stack Command Subpacket

Offset	Contents
?ZSKI (double word)	Subpacket identifier. Place ?ZSKZ here.
?ZSKF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZSK.
	Reserved. (Set to 0.)
?ZSK3	Reserved. (Set to 0.)
?ZSKR (double word)	Reserved. (Set to 0.)

Start Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device or cooperative name buffer for the start function.

Figure 2-151 shows the structure of the start command subpacket, and Table 2-129 describes its contents.

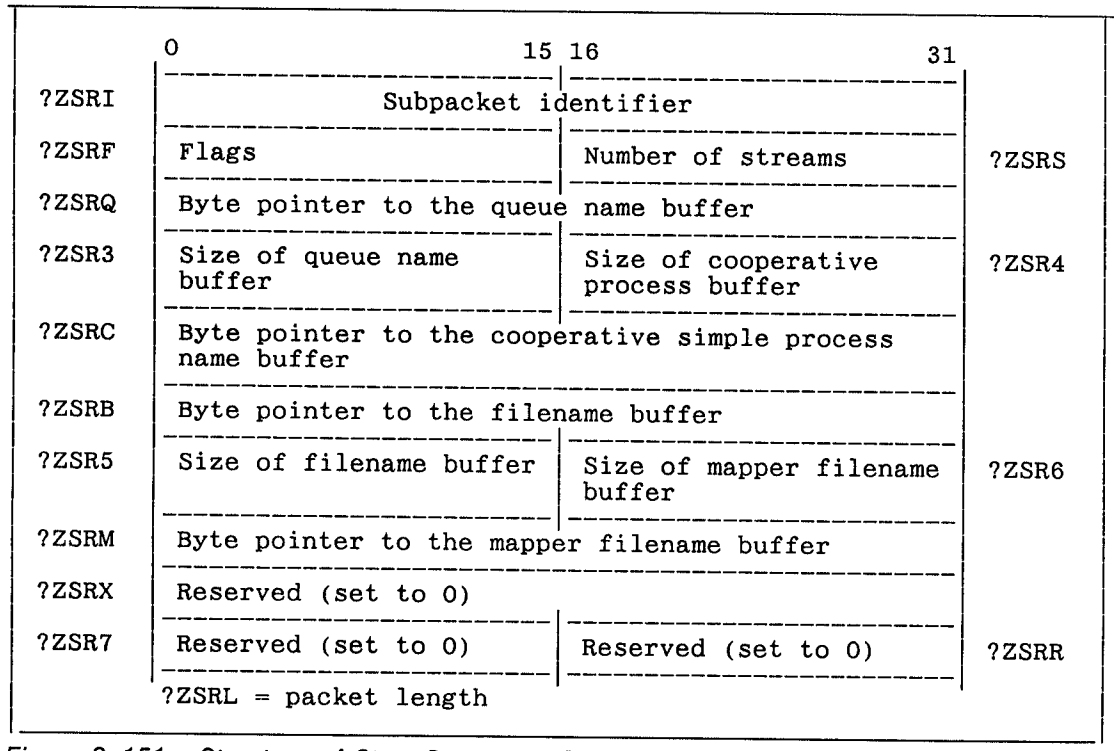


Figure 2-151. Structure of Start Command Subpacket

Table 2-129. Contents of Start Command Subpacket

Offset	Contents
?ZSRI (double word)	Subpacket identifier. Place ?ZSRZ here.
?ZSRF	Flags word. It contains bit flags with additional information to the operating system and EXEC for function ?ZSR. ?ZYU0 = 1B(?ZZU0) -- New line switch. ?ZYU1 = 1B(?ZZU1) -- 8-bit switch. ?ZYU2 = 1B(?ZZU2) -- Process switch. ?ZYU3 = 1B(?ZZU3) -- Streams switch. ?ZYU4 = 1B(?ZZU4) -- Name switch. ?ZYU5 = 1B(?ZZU5) -- Upper argument specified. ?ZYU6 = 1B(?ZZU6) -- Reserved (Set to 0.) ?ZYU7 = 1B(?ZZU7) -- IPC switch.
?ZSRS	Specify the number of streams.
?ZSRQ (double word)	Supply a byte pointer to the queue name buffer.
?ZSR3	Supply the number of bytes in the queue name buffer.
?ZSR4	Supply the number of bytes in the cooperative simple process name buffer.
?ZSRC (double word)	Supply a byte pointer to the cooperative simple process name buffer.
?ZSRB (double word)	Supply a byte pointer to the IPC filename buffer.
?ZSR5	Supply the number of bytes in the IPC filename buffer.
?ZSR6	Supply the number of bytes in the mapper filename buffer.
?ZSRM (double word)	Supply a byte pointer to the mapper filename buffer.
?ZSRX (double word)	Reserved. (Set to 0.)
?ZSR7	Reserved. (Set to 0.)
?ZSRR	Reserved. (Set to 0.)

Status Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the status function. To obtain the status of all default batch streams and have the queue name returned in the main packet, set bit ?ZZV0 of offset ?ZSSF of the subpacket.

Figure 2-152 shows the structure of the status command subpacket, and Table 2-130 describes its contents.

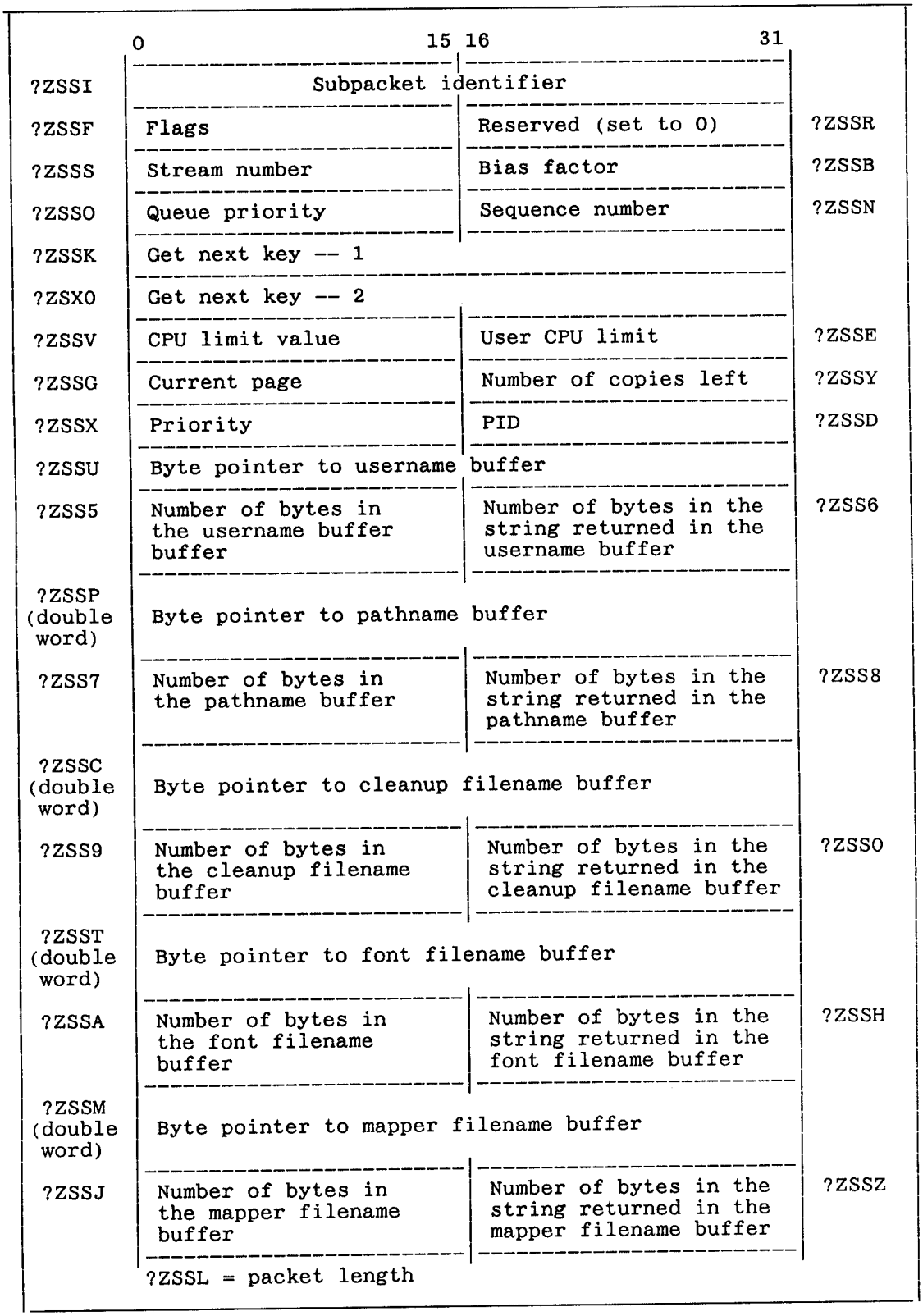


Figure 2-152. Structure of Status Command Subpacket

Licensed Material - Property of Data General Corporation

Table 2-130. Contents of Status Command Subpacket

Offset	Contents
?ZSSI (double word)	Subpacket identifier. Place ?ZQSS here.
?ZSSF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYV0 = 1B(?ZZV0) -- You want the status of all default batch input streams. ?ZJV1 = 0B(?ZZV1) -- An output bit; stream/device is idle. ?ZYV1 = 1B(?ZZV1) -- An output bit; stream/device is active. ?ZYV2 = 1B(?ZZV2) -- An output bit; process type is swappable. ?ZYV3 = 1B(?ZZV3) -- An output bit; process type is pre-emptible. ?ZYV4 = 1B(?ZZV4) -- An output bit; process type is is resident. ?ZJV5 = 0B(?ZZV5) -- An output bit; 7-bit print mode. ?ZYV5 = 1B(?ZZV5) -- An output bit; 8-bit print mode. ?ZJV6 = 0B(?ZZV6) -- An output bit; stream/device is not ready. ?ZYV6 = 1B(?ZZV6) -- An output bit; stream/device is ready. ?ZYV7 = 1B(?ZZV7) -- An output bit; stream/device is paused. (If the active bit is also set, the stream/device will be paused.) ?ZYV8 = 1B(?ZZV8) -- An output bit; limiting is enabled. ?ZYV9 = 1B(?ZZV9) -- An output bit; device is being aligned. ?ZYVA = 1B(?ZZVA) -- An output bit; batch job. ?ZYVB = 1B(?ZZVB) -- An output bit; print or plot device. ?ZYVC = 1B(?ZZVC) -- An output bit; net cooperative. ?ZJVD = 0B(?ZZVD) -- An output bit; net cooperative is not ready. ?ZYVD = 1B(?ZZVD) -- An output bit; net cooperative is ready. ?ZJVE = 0B(?ZZVE) -- An output bit; net cooperative is idle. ?ZYVE = 1B(?ZZVE) -- An output bit; net cooperative is active.

(Continues)

Table 2-130. Contents of Status Command Subpacket

Offset	Contents
?ZSSR	Reserved. (Set to 0.)
?ZSSS	Supply the stream number.
?ZSSB	The operating system returns the bias factor.
?ZSSO	The operating system returns the queue priority.
?ZSSN	The operating system returns the sequence number.
?ZSSK (double word)	Get next key entry. The operating system uses this offset to hold the key identifier for EXEC.
?ZSX0 (double word)	Second get next key entry. The operating system uses this offset to hold the second key indicator for EXEC.
?ZSSV	The operating system returns the system's CPU limit value.
?ZSSE	The operating system returns the user-specified CPU limit value.
?ZSSG	The operating system returns the current page.
?ZSSY	The operating system returns the number of copies left.
?ZSSX	The operating system returns the priority.
?ZSSD	The operating system returns the PID.
?ZSSU (double word)	Supply a byte pointer to the buffer to receive the username.
?ZSS5	Specify the number of bytes in the username buffer.
?ZSS6	The operating system returns the number of bytes in the string that it placed in the username buffer.
?ZSSP (double word)	Supply a byte pointer to the buffer that receives the pathname.
?ZSS7	Specify the number of bytes in the pathname buffer.
?ZSS8	The operating system returns the number of bytes in the string that it placed in the pathname buffer.
?ZSSC (double word)	Supply a byte pointer to the buffer that receives the cleanup filename.
?ZSS9	Specify the number of bytes in the cleanup filename buffer.

(Continued)

Table 2-130. Contents of Status Command Subpacket

Offset	Contents
?ZSSO	The operating system returns the number of bytes in the string that it placed in the cleanup filename buffer.
?ZSST (double word)	Supply a byte pointer to the buffer that receives the font filename.
?ZSSA	Specify the number of bytes in the font filename buffer.
?ZSSH	The operating system returns the number of bytes in the string that it placed in the font filename buffer.
?ZSSM (double word)	Supply a byte pointer to the buffer that receives the mapper filename.
?ZSSJ	Specify the number of bytes in the mapper filename buffer.
?ZSSZ	The operating system returns the number of bytes in the string that it placed in the mapper filename buffer.

(Concluded)

Stop Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the stop function.

Figure 2-153 shows the structure of the stop command subpacket, and Table 2-131 describes its contents.

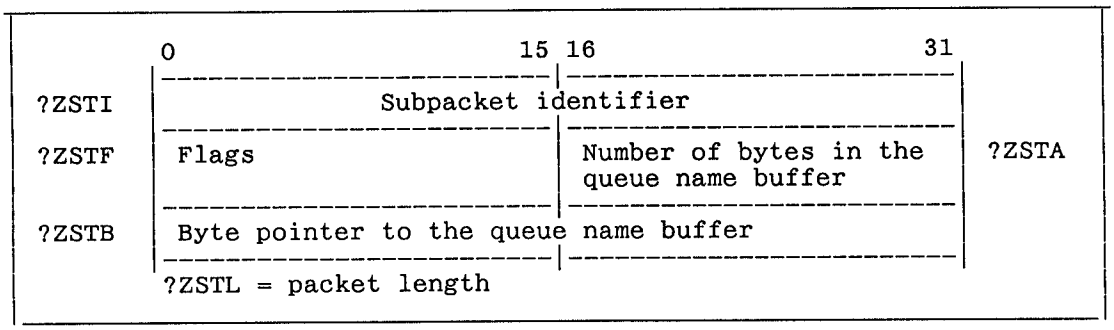


Figure 2-153. Structure of Stop Command Subpacket

Table 2-131. Contents of Stop Command Subpacket

Offset	Contents
?ZSTI (double word)	Subpacket identifier. Place ?ZSTZ here.
?ZSTF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYWO = 1B(?ZZWO) -- Queue name specified.
?ZSTA	Specify the number of bytes in the queue name buffer if you need the buffer.
?ZSTB (double word)	Supply a byte pointer to the queue name buffer if you need the buffer.

Licensed Material - Property of Data General Corporation

Terminate Command

This function does not require a subpacket. Instead, supply a byte pointer to the device name buffer in offset ?ZXNA of the ?OPEX main packet in Figure 2-111.

Licensed Material - Property of Data General Corporation

Trailers Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the trailers function.

Figure 2-154 shows the structure of the trailers command subpacket, and Table 2-132 describes its contents.

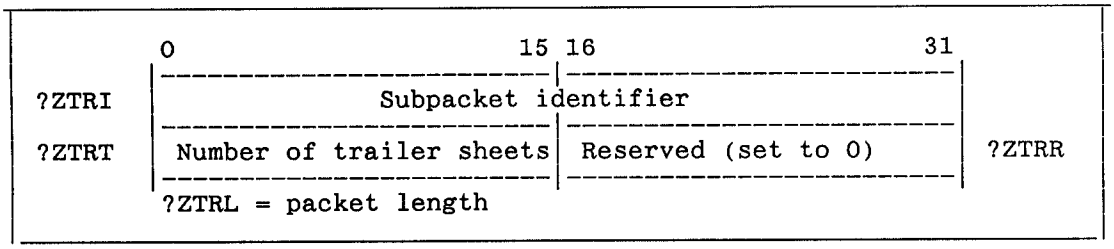


Figure 2-154. Structure of Trailers Command Subpacket

Table 2-132. Contents of Trailers Command Subpacket

Offset	Contents
?ZTRI (double word)	Subpacket identifier. Place ?ZTRZ here.
?ZTRT	Supply the number of trailer sheets after each printed file.
?ZTRR	Reserved. (Set to 0.)

Licensed Material - Property of Data General Corporation

Unhold Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the username buffer, if necessary, for the unhold function.

Figure 2-155 shows the structure of the unhold command subpacket, and Table 2-133 describes its contents.

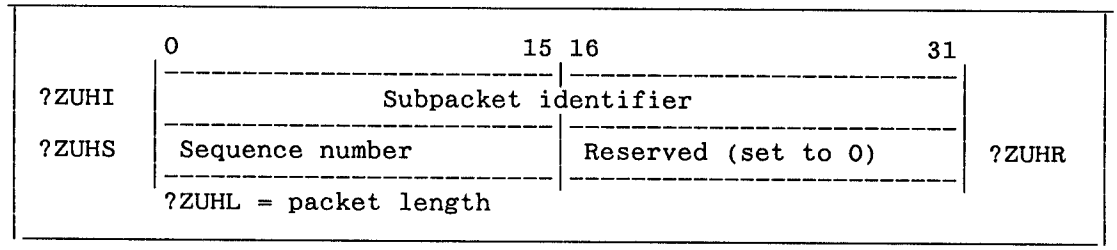


Figure 2-155. Structure of Unhold Command Subpacket

Table 2-133. Contents of Unhold Command Subpacket

Offset	Contents
?ZUHI (double word)	Subpacket identifier. Place ?ZUHZ here.
?ZUHS	Supply the sequence number in this offset.
?ZUHR	Reserved. (Set to 0.)

Unitstatus Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device name buffer for the unitstatus function.

Figure 2-156 shows the structure of the unitstatus command subpacket, and Table 2-134 describes its contents.

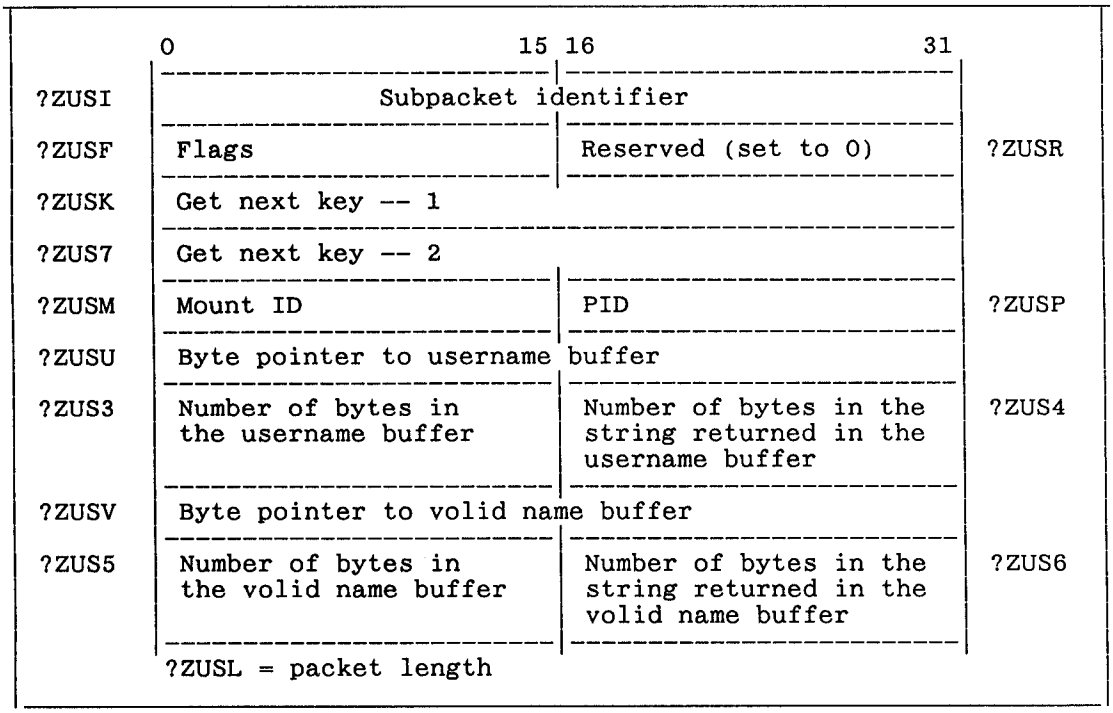


Figure 2-156. Structure of Unitstatus Command Subpacket

Table 2-134. Contents of Unitstatus Command Subpacket

Offset	Contents
?ZUSI (double word)	Subpacket identifier. Place ?ZUSZ here.
?ZUSF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZY70 = 1B(?ZZ70) -- All units. ?ZY71 = 1B(?ZZ71) -- An output bit; unit not mounted. ?ZY72 = 1B(?ZZ72) -- An output bit; unit is premounted. ?ZY73 = 1B(?ZZ73) -- An output bit; unit is waiting to be dismounted. ?ZY74 = 1B(?ZZ74) -- An output bit; IBM format.
?ZUSR	Reserved. (Set to 0.)
?ZUSK (double word)	Get next entry key. The operating system uses this offset to hold the key identifier for EXEC.
?ZUS7 (double word)	Second get next entry key. The operating system uses this offset to hold the second key indicator for EXEC.
?ZUSM	The operating system returns the mount ID.
?ZUSP	The operating system returns the PID.
?ZUSU (double word)	Supply a byte pointer to the buffer that receives the username.
?ZUS3	Specify a number of bytes to accommodate the largest possible username.
?ZUS4	The operating system returns the number of bytes in the username that it placed in the username buffer.
?ZUSV (double word)	Supply a byte pointer to the buffer that receives the volid.
?ZUS5	Specify a number of bytes to accommodate the largest possible volid.
?ZUS6	The operating system returns the number of bytes in the volid that it placed in the volid buffer.

Unlimit Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the unlimit function. If you take the default option and set flag bit ?ZZX0 in offset ?ZULF of the subpacket, then the unlimit function applies to all batch streams.

Figure 2-157 shows the structure of the unlimit command subpacket, and Table 2-135 describes its contents.

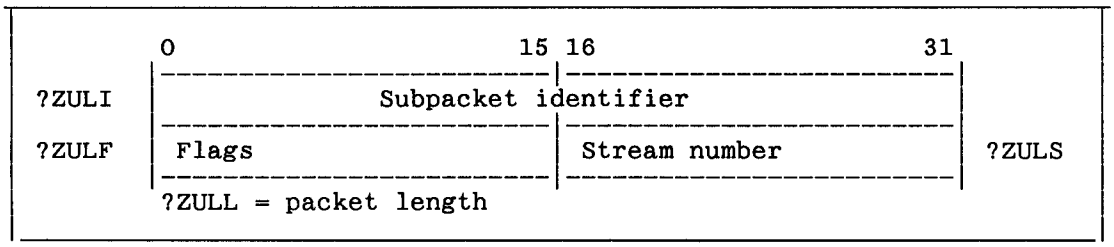


Figure 2-157. Structure of Unlimit Command Subpacket

Table 2-135. Contents of Unlimit Command Subpacket

Offset	Contents
?ZULI (double word)	Subpacket identifier. Place ?ZULZ here.
?ZULF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYX0 = 1B(?ZZX0) -- Unlimit all default batch input streams.
?ZULS	Supply the number of the stream if bit ?ZZX0 in offset ?ZULF is off. Otherwise, supply 0.

Unsilence Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the unsilence function. If you take the default option and set flag bit ?ZZY0 in offset ?ZUNF of the subpacket, then the unsilence function applies to all batch streams.

Figure 2-158 shows the structure of the unsilence command subpacket, and Table 2-136 describes its contents.

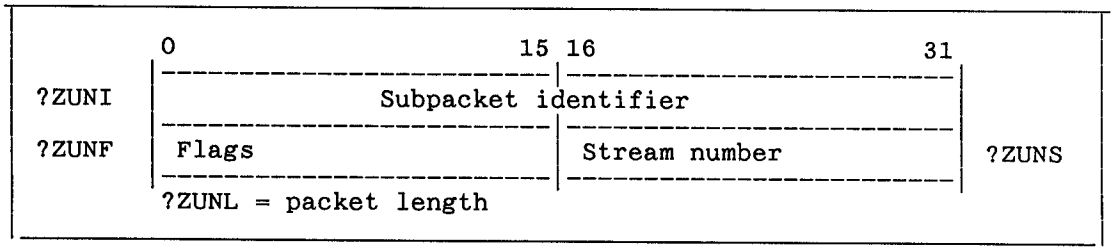


Figure 2-158. Structure of Unsilence Command Subpacket

Table 2-136. Contents of Unsilence Command Subpacket

Offset	Contents
?ZUNI (double word)	Subpacket identifier. Place ?ZUNZ here.
?ZUNF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYY0 = 1B(?ZZY0) -- Unsilence all default batch input streams.
?ZUNS	Supply the number of the stream if bit ?ZZY0 in offset ?ZUNF is off. Otherwise, supply 0.

Verbose Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the verbose function. If you take the default option and set flag bit ?ZZZ0 in offset ?ZVEF of the subpacket, then the verbose function applies to all batch streams.

Figure 2-159 shows the structure of the verbose command subpacket, and Table 2-137 describes its contents.

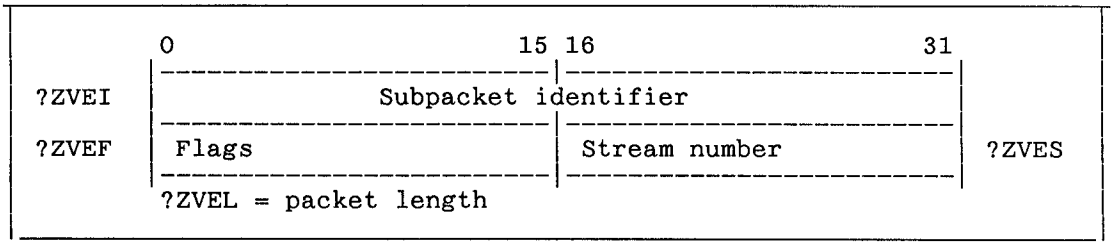


Figure 2-159. Structure of Verbose Command Subpacket

Table 2-137. Contents of Verbose Command Subpacket

Offset	Contents
?ZVEI (double word)	Subpacket identifier. Place ?ZVEZ here.
?ZVEF	Flags word. It contains bit flags that indicate default choices to the operating system. ?ZYZ0 = 1B(?ZZZ0) -- Verbose on all default batch input streams.
?ZVES	Supply the number of the stream if bit ?ZZZ0 in offset ?ZVEF is off. Otherwise, supply 0.

Xbias Command

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the device/queue name buffer for the xbias function.

There are two correspondences between the CLI syntax for the xbias command and the ?OPEX equivalent. They are next.

```
CONTROL @EXEC XBIAS    [QUEUENAME]          [STREAM #]    [BIAS FACTOR]
?OPEX         FUNC     (OPEX_PACKET.NAME)   ARGUMENT_1    ARGUMENT_2

CONTROL @EXEC XBIAS    @DEVICENAME          [BIAS FACTOR]
?OPEX         FUNC     (OPEX_PACKET.NAME)   ARGUMENT_1
```

Figure 2-160 shows the structure of the xbias command subpacket, and Table 2-138 describes its contents.

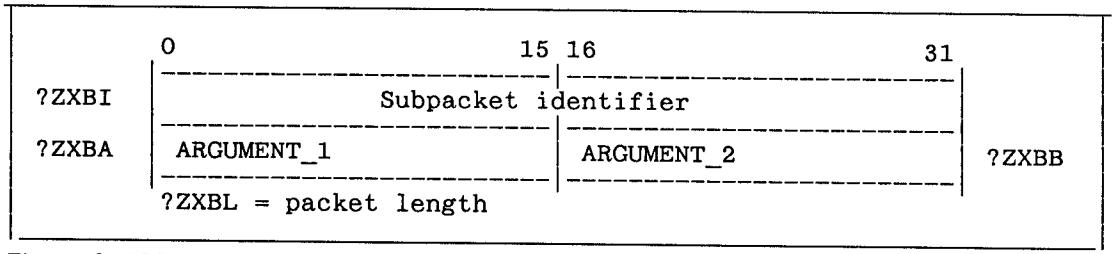


Figure 2-160. Structure of Xbias Command Subpacket

Table 2-138. Contents of Xbias Command Subpacket

Offset	Contents
?ZXBI (double word)	Subpacket identifier. Place ?ZXBZ here.
?ZXBA	ARGUMENT_1. If you specify a queue name, supply a stream number here. The default value is all batch streams; to specify this, supply -1. If you specify a device name, then supply the bias factor.
?ZXBB	ARGUMENT_2. If you specify a queue name, supply the bias factor. If you specify a device name, then the operating system ignores this offset. (Set it to 0.)

Autokey Command

Only AOS/DVS supports this function.

Bit ?ZZA0 of offset ?ZXFG of the main ?OPEX packet in Figure 2-111 must not be set, since the autokey function does not require a buffer to pass information. Consequently, the operating system ignores offset ?ZXNA of the main ?OPEX packet.

There is a correspondence between the CLI syntax for the autokey command and the ?OPEX equivalent. It is next.

```
CONTROL @EXEC AUTOKEY                          [ARGUMENT_1] [ARGUMENT_2]
?OPEX         FUNC     (OPEX_PACKET.NAME)   ARGUMENT_1   ARGUMENT_2
```

Figure 2-161 shows the structure of the autokey command subpacket, and Table 2-139F describes its contents.

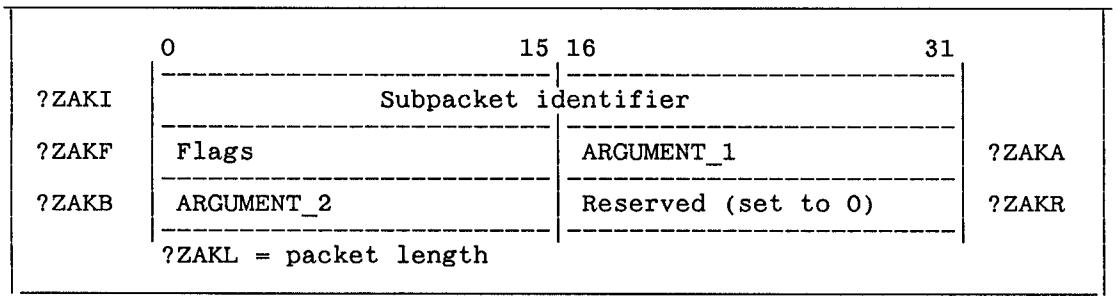


Figure 2-161. Structure of Autokey Command Subpacket

Table 2-139. Contents of Autokey Command Subpacket

Offset	Contents
?ZAKI (double word)	Subpacket identifier. Place ?ZAKZ here.
?ZAKF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZAK. ?ZJAF = 0B(?ZZAF) -- You want to set the autokey sequence. ?ZYAF = 1B(?ZZAF) -- The default. You want the status of the current autokey setting. ?ZJAG = 0B(?ZZAG) -- You want to set a single-character autokey sequence. ?ZYAG = 1B(?ZZAG) -- You want to set a two-character autokey sequence. ?ZJAH = 0B(?ZZAH) -- An output bit; the current autokey status is a single-character. ?ZYAH = 1B(?ZZAH) -- An output bit; the current autokey status is two characters.
?ZAKA	ARGUMENT_1. Supply the first autokey character.
?ZAKB	ARGUMENT_2. Supply the optional second autokey character.
?ZAKR	Reserved. (Set to 0.)

Register Command

Only AOS/DVS supports this function.

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the queue name you want to register.

The CLI syntax for the register command is next.

```
CONTROL @EXEC REGISTER QUEUENAME [REGISTERED_QUEUENAME]
```

Figure 2-162 shows the structure of the register command subpacket, and Table 2-140 describes its contents.

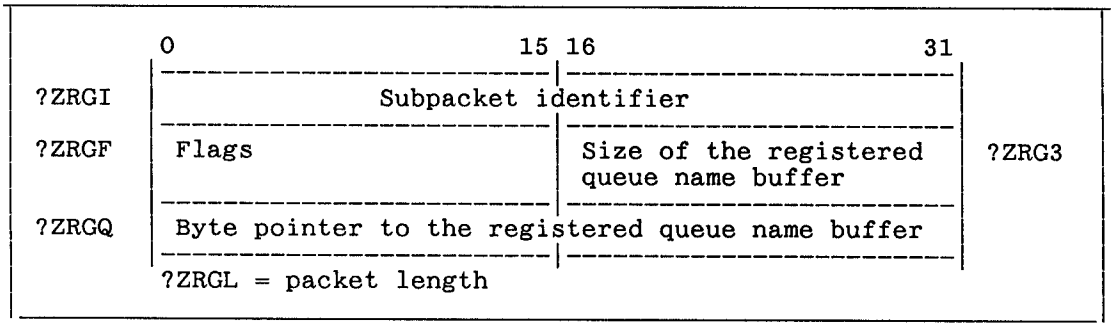


Figure 2-162. Structure of Register Command Subpacket

Table 2-140. Contents of Register Command Subpacket

Offset	Contents
?ZRGI (double word)	Subpacket identifier. Place ?ZRGZ here.
?ZRGF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZRG. ?ZYBF = 1B(?ZZBF) -- If set, the bytepointer to the registered queue name buffer is in offset ?ZRGQ.
?ZRG3	Supply the number of bytes in the registered queue name buffer.
?ZRGQ (double word)	Supply the byte pointer to the registered queue name buffer.

Deregister Command

Only AOS/DVS supports this function.

This function does not require a subpacket. Instead, supply a byte pointer to the registered queue name buffer in offset ?ZXNA of the ?OPEX main packet in Figure 2–111.

The CLI syntax for the register command is next.

```
CONTROL @EXEC DEREGISTER REGISTERED_QUEUENAME
```

Registerstatus Command

Only AOS/DVS supports this function.

Offset ?ZXNA of the main ?OPEX packet in Figure 2-111 must contain the byte address of the local queue name buffer for the registerstatus function.

The CLI syntax for the registerstatus command is next.

```
CONTROL @EXEC REGISTERSTATUS [LOCAL_QUEUENAME]
```

Figure 2-163 shows the structure of the registerstatus command subpacket, and Table 2-141 describes its contents.

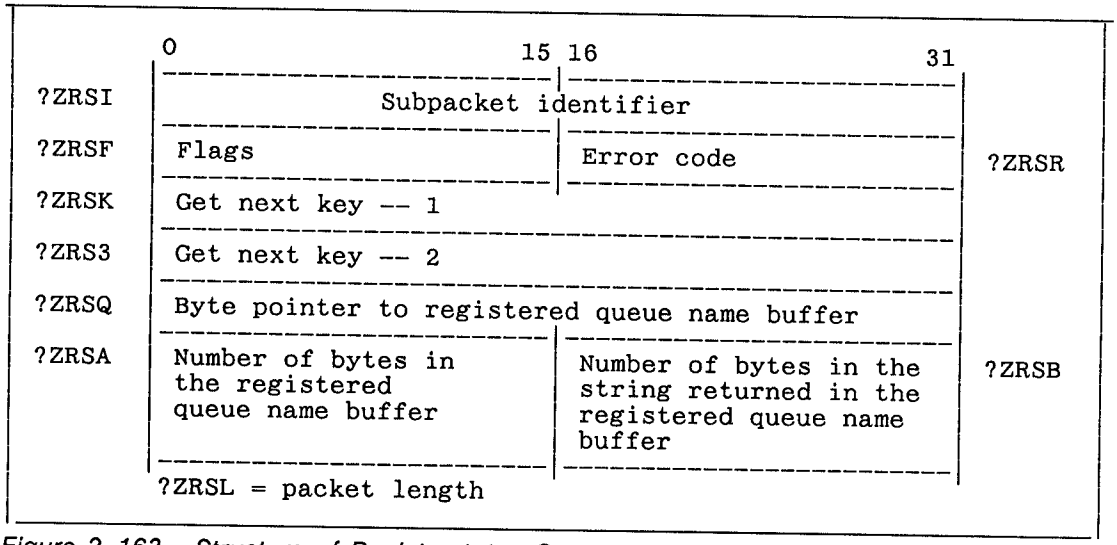


Figure 2-163. Structure of Registerstatus Command Subpacket

Table 2-141. Contents of Registerstatus Command Subpacket

Offset	Contents
?ZRSI (double word)	Subpacket identifier. Place ?ZRSZ here.
?ZRSF	Flags word. It contains the bit flags with additional information to the operating system and EXEC for function ?ZRS. ?ZYCF = 1B(?ZZCF) -- The default. You want the status of all local queues. The operating system returns their names in the local queue name buffer that offset ?ZRSQ points to.
?ZRSR	EXEC returns an error code to this offset.
?ZRSK (double word)	Get next entry key. The operating system uses this offset to hold the key indicator for EXEC.
?ZRS3 (double word)	Second get next entry key. The operating system uses this offset to hold the second key indicator for EXEC.
?ZRSQ (double word)	Supply a byte pointer to the buffer that receives the registered queue name.
?ZRSA	Specify a number of bytes to accommodate the largest possible registered queue name.
?ZRSB	The operating system returns the number of bytes in the registered queue name that it placed in the registered queue name buffer.

?OVEX

Releases an overlay and returns. (16–Bit Processes Only)

?OVEX
error return

Input

AC0 Contains the following:

- Bit 0 is ignored
- Overlay area number (defined in the .ENTO argument) in Bits 1 through 6
- Overlay number (defined in the .ENTO argument) in Bits 7 through 15

AC1 Reserved (Set to 0.)

AC2 Normal return address

Output

None

Error Codes in AC0

ERICM	Illegal system command
ERROO	Invalid overlay number
EROVN	Illegal overlay number

Why Use It?

?OVEX is one of the three primitive overlay system calls that release an overlay. (The others are ?OVREL and ?OVKIL.) If you load an overlay with the primitive overlay system call ?OVLOD, you must use ?OVEX, ?OVREL, or ?OVKIL to release it.

?OVEX offers two advantages over the other release system calls: you can issue ?OVEX from within an overlay (unlike ?OVREL), and you can use ?OVEX to release an overlay without killing the calling task (unlike ?OVKIL).

What It Does

?OVEX exits from an overlay loaded by a previous ?OVLOD system call, decrements the overlay's overlay use count (OUC), and frees the overlay area if the OUC becomes 0. ?OVEX also transfers control to an address outside the overlay. This can be the caller's return address, if the caller is returning from a subroutine within the overlay.

Note that you must specify the ?OVEX return address in AC2.

Notes

- See the descriptions of ?OVLOD, ?OVREL, ?OVKIL in this chapter.

Licensed Material - Property of Data General Corporation

?OVKIL Exits from an overlay and kills the calling task. (16–Bit Processes Only)

```
?OVKIL
error return
```

Input

AC0 Contains the following:

- Bit 0 is ignored
- Overlay area number (defined in the .ENTO argument) in Bits 1 through 6
- Overlay number (defined in the .ENTO argument) in Bits 7 through 15

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

None

Error Codes in AC0

ERICM	Illegal system command
ERROO	Invalid overlay number
EROVN	Illegal overlay number

Why Use It?

?OVKIL is one of the three primitive system calls for releasing an overlay. (The others are ?OVEX and ?OVREL). Use ?OVKIL if you not only want to release an overlay, but you also want to kill the task using the overlay and transfer control to the task with the next highest priority.

What It Does

?OVKIL decrements the target overlay's overlay use count (OUC) value, releases the overlay if the OUC becomes 0, and kills the calling task. Because ?OVKIL kills the calling task, there is no normal return for ?OVKIL.

Notes

- See the descriptions of ?OVEX and ?OVREL in this chapter.

?OVLOD

Loads and goes to an overlay. (16–Bit Processes Only)

?OVLOD
error return
normal return

Input

AC0 Contains the following:

- Bit 0 is a flag bit:

  Bit 0 = 0 on a conditional load

  Bit 0 = 1 on an unconditional load

- Overlay area number (defined in the .ENTO argument) in Bits 1 through 6
- Overlay number (defined in the .ENTO argument) in Bits 7 through 15

AC1 Contains the following:

- Bit 0 is a flag bit:

  Bit 0 = 0 to pass control to an offset within the overlay

  Bit 0 = 1 to pass control to an address within the overlay

- Target address or offset within the overlay (If Bits 1 through 15 = –1, control returns to the ?OVLOD normal return.)

AC2 Contents passed to the overlay (unless AC1 contains –1)

Output

AC0 Unchanged

AC1 Base address of the target overlay

AC2 Unchanged

Licensed Material – Property of Data General Corporation

Error Codes in AC0

ERADR	Illegal starting address
ERICM	Illegal system command
EROVN	Illegal overlay number

Why Use It?

?OVLOD is an alternate way of calling an overlay from the .OL file to an overlay area in memory. ?OVLOD and the other primitive overlay system calls give you direct control of overlay management, but they also require that you explicitly load and release each overlay. The resource system calls (?RCALL, ?KCALL, and ?RCHAIN) manage overlays automatically. Therefore, we recommend that you use resource system calls instead of primitive overlay system calls.

What It Does

?OVLOD loads an overlay into its overlay area and, optionally, transfers control to some point in the overlay (which you specify as an address or as an offset).

To use ?OVLOD and the other primitive overlay system calls (?OVREL, ?OVEX, and ?OVKIL), you must define each overlay with the .ENTO (overlay entry) pseudo-op.

The input status of Bit 0 in AC0 governs whether the operating system loads the overlay conditionally or unconditionally. If you specify conditional loading (Bit 0 of AC0 = 0), the operating system loads the overlay only if it is not already loaded. If the overlay is resident, a conditional ?OVLOD simply directs the operating system to increment the overlay's OUC value. If you specify unconditional loading, the operating system loads the overlay, whether or not it is resident in the overlay area.

If you set Bits 1 through 15 of AC1 to -1, ?OVLOD takes the normal return after the operating system loads the overlay. Otherwise, the operating system interprets AC1 as follows:

- If Bit 0 of AC1 = 1, then the operating system treats Bits 1 through 15 as an offset into the overlay and passes control to that offset.
- If Bit 0 of AC1 = 0, then the operating system treats Bits 1 through 15 as an absolute address in the overlay, and passes control to that address.

?OVLOD Continued

Note that you must set Bits 1 through 15 in AC1 to an offset if the total overlay area is a multiple of the basic overlay area. You cannot specify an absolute address in this case, because the operating system may relocate the overlay to any basic area within the total area. Thus, any absolute address you give could be invalid.

If you transfer control to the overlay (rather than the ?OVLOD normal return), the operating system passes the overlay the base address in AC1 after it executes ?OVLOD. If the overlay then issues ?OVKIL, the operating system never takes the ?OVLOD normal return, because ?OVKIL kills the calling task.

The caller receives the overlay base address (and the other output values) only if AC1 contains −1 on input or if the loaded overlay returns control to the caller with an RTN instruction. Otherwise, the output values at the ?OVLOD normal return are those passed as input values to ?OVEX.

Notes

- See the description of ?OVKIL in this chapter.

?OVREL

Releases an overlay area. (16–Bit Processes Only)

?OVREL
error return
normal return

Input

AC0	Contains the following: • Bit 0 is ignored • Overlay area number (defined in the .ENTO argument) in Bits 1 through 6 • Overlay number (defined in the .ENTO argument) in Bits 7 through 15
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERICM	Illegal system command
ERROO	Invalid overlay number
EROVN	Illegal overlay number

Why Use It?

?OVREL is one of the three primitive system calls for releasing an overlay. (The others are ?OVEX and ?OVLOD). Use ?OVREL if you want to release an overlay and maintain the calling task in the active state. (?OVKIL kills the calling task.)

What It Does

?OVREL decrements the overlay use count (OUC) of an overlay previously loaded with ?OVLOD and frees the overlay area if the OUC becomes 0. You cannot issue ?OVREL from the overlay you want to release, because the normal return, in this case, would be in the target overlay.

Notes

- See the descriptions of ?OVLOD, ?OVEX, ?OVKIL in this chapter.

Licensed Material - Property of Data General Corporation

?PCLASS

Gets a process's class and locality.

AOS/VS only

?PCLASS *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?PCLASS packet

Error Codes in AC0

ERPRH	Attempt to access process not in hierarchy

Why Use It?

Use this system call to obtain locality and class information for a specific process.

What It Does

This system call accepts a process identifier from you. It returns the process's user locality, its program locality, and its class information.

If you want information about the calling process, place the value of ?PCL_PID in offset ?PCL_PKT.PCODE of the packet and place −1 in offset ?PCL_PKT.PID.

Licensed Material - Property of Data General Corporation

You obtain information about a different process in one of the two following ways:

- Assume that you know the name of the process. Place the value of ?PCL_PNAME in offset ?PCL_PKT.PCODE. Place in offset ?PCL_PKT.PNAME a byte pointer to the process name. Set offset ?PCL_PKT.PID to zero.
- Assume that you know the PID of the process. Place the value of ?PCL_PID in offset ?PCL_PKT.PCODE. Place zero in offset ?PCL_PKT.PNAME. Place the PID in offset ?PCL_PKT.PID.

Figure 2-164 shows the structure of the ?PCLASS parameter packet, and Table 2-142 describes its contents.

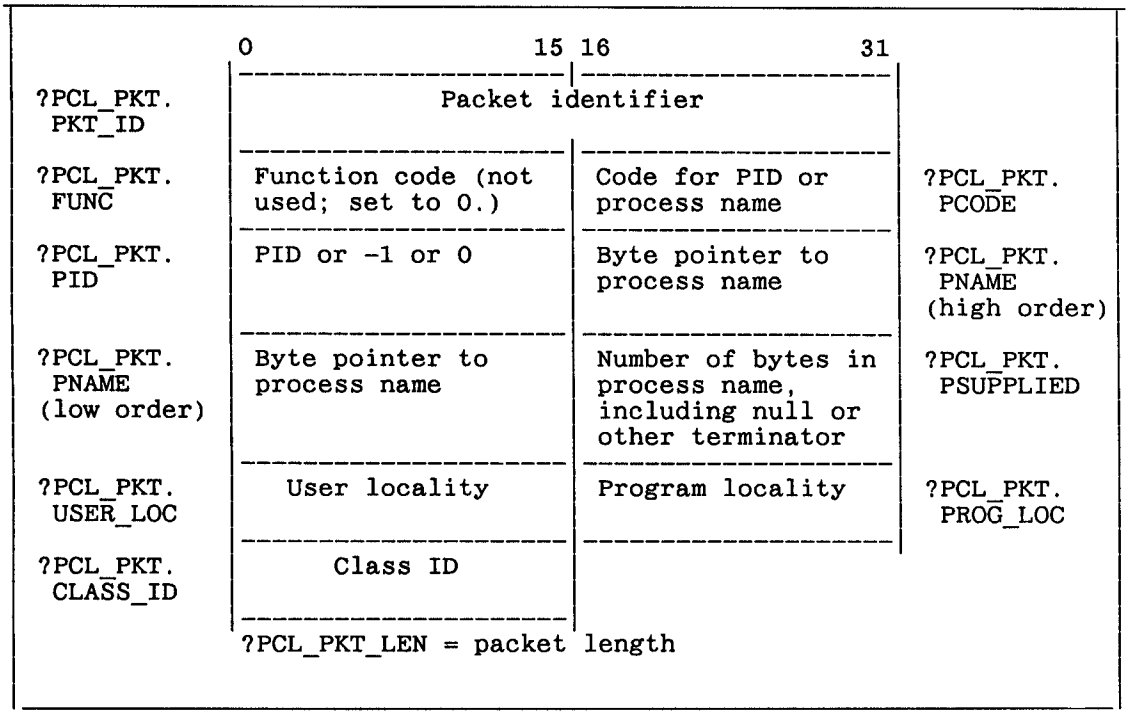


Figure 2-164. Structure of ?PCLASS Packet

?PCLASS Continued

Table 2-142. Contents of ?PCLASS Packet

Offset	Contents
?PCL_PKT.PKT_ID (double word)	Packet identifier. Place ?PCL_PKT_PKTID here.
?PCL_PKT.FUNC	Function code. Not used. (Set to 0.)
?PCL_PKT.PCODE	Code word into which you place ?PCL_PID when you supply a PID, or else ?PCL_PNAME when you supply a process name.
?PCL_PKT.PID	Supply -1 for the calling process. For another process, either place its PID here, or else place 0 here and the process's name information in the next two offsets.
?PCL_PKT.PNAME (double word)	If you've placed 0 in the previous offset, then place a byte pointer here to the process's name, and place its length in the next offset.
?PCL_PKT.PSUPPLIED	If you've placed 0 in offset ?PCL_PKT.PID and a byte pointer in the previous offset, then place the number of bytes in the byte pointer here.
?PCL_PKT.USER_LOC	User locality.
?PCL_PKT.PROG_LOC	Program locality.
?PCL_PKT.CLASS_ID	Class ID.

Licensed Material - Property of Data General Corporation

?PCNX — Passes a connection from one server to another in Ring 7.

?PCNX
error return
normal return

Input

AC0	Customer's PID
AC1	PID of the new server
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection does not exist
ERCCS	Cannot connect to self
ERNAS	Process is not a server
ERPRH	Attempt to access process not in hierarchy

Why Use It?

?PCNX allows you to break a connection between a server and one of its customers and to re-establish the connection with a new server process. ?PCNX is useful for passing a customer/server connection from an intermediate server to a specialized server.

What It Does

?PCNX passes the connection established between the caller and one of its customers to another server process. Before you issue ?PCNX, load AC0 with the PID of the customer process, and load AC1 with the PID of the new server. Note that AC1 cannot contain the PID of the calling process.

The calling process must be a server of the customer that you specify in AC0, and there must be an unbroken connection between the two. The new server process must have already issued ?SERVE to declare itself a server.

When a server issues ?PCNX, the operating system updates the connection-table entry so that it contains the PID of the new server.

?PGLIST Gets a process group name search list.

AOS/DVS only

?PGLIST
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

Why Use It?

?PGLIST lets you examine the current contents of your process group name search list. Thus, you can use ?PGLIST as a preliminary system call to ?PSLIST, which changes the contents of the process group name search list.

Notes

- See also the description of ?GLIST, which deals with filename search lists.

Licensed Material - Property of Data General Corporation

?PGNAME — Returns a process group's identifier.

AOS/DVS only

?PGNAME *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?PGNAME packet

Error Codes in AC0

ERFOR	Function out of range
ERHNE	Invalid process group identifier
ERPGN	Process group does not exist

Why Use It?

Use this system call to obtain the process group ID (PGID) given the name of a process group (PGNAME), or else to obtain the name of a process group given its PGID.

For example, a process that wishes to identify itself unambiguously can use a fully qualified process name, a unique PID, or a local PID and PGID pair. The third method is consistent with the functionality provided by the ?TPID and ?GVPID system calls. In order to use these calls in the AOS/DVS environment, a process would issue ?PGNAME to obtain its own PGID.

What It Does

On each AOS/DVS node, processes are organized into one or more aggregates known as process groups. Each process group is identified by both a text name (which must be registered and thus unique within a community) and by a 16-bit process group identifier — a PGID.

?PGNAME Continued

?PGNAME returns a process group's ID number from its name, and vice-versa. You can also use it to determine both the name and PGID of the process group of which your process (the one issuing ?PGNAME) is a member.

Note that this call clearly distinguishes between operating system defined resources (process groups) and networking defined resources (XODIAC™ hosts).

Figure 2-165 shows the structure of ?PGNAME's parameter packet and Table 2-143 describes its contents.

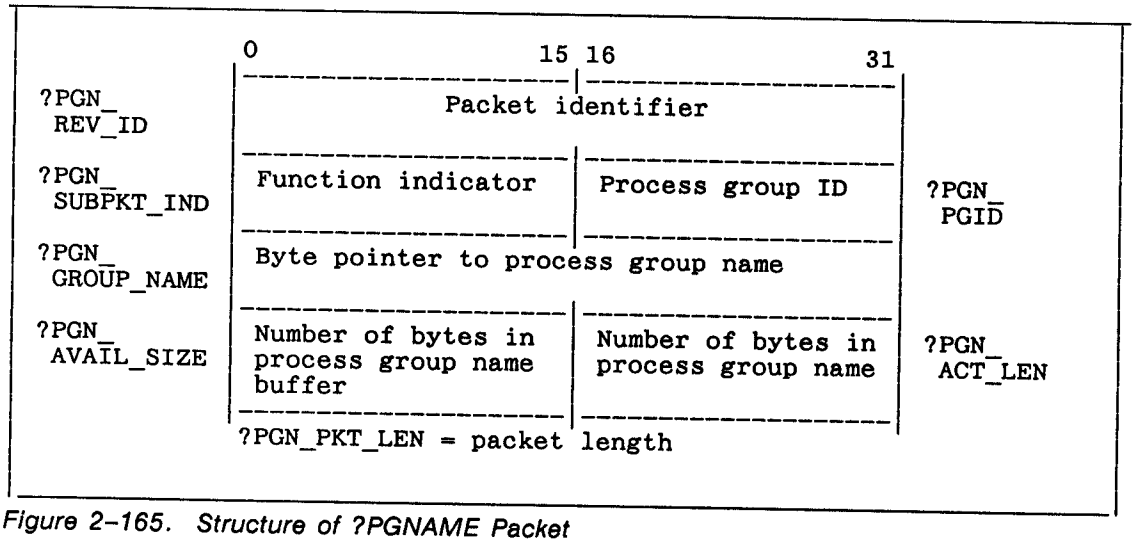


Figure 2-165. Structure of ?PGNAME Packet

Licensed Material - Property of Data General Corporation

Table 2-143. Contents of ?PGNAME Packet

Offset	Contents
?PGN_REV_ID (double word)	Packet identifier. Supply ?PGN_PKTID.
?PGN_SUBPKT_IND	Function code. Your supplied values have the following effects: ?PGLID -- return local PGID. ?PGLNM -- return local PGNAME. ?PGBIN -- return both local PGID and PGNAME. ?PGRID -- return PGID of a given PGNAME. ?PGRNM -- return PGNAME of a given PGID.
?PGN_PGID	If offset ?PGN_SUBPKT_IND contains — then in offset ?PGN_PGID ?PGLID — AOS/DVS returns the local PGID. ?PGBIN — AOS/DVS returns the local PGID. ?PGLNM — Undefined; set to 0. ?PGRID — AOS/DVS returns the PGID of the given PGNAME. ?PGRNM — AOS/DVS returns the PGNAME of the given PGID.
?PGN_GROUP_NAME (double word)	If offset ?PGN_SUBPKT_IND contains — then in offset ?PGN_GROUP_NAME ?PGLID — Undefined; set to 0. ?PGBIN — AOS/DVS returns a byte pointer to the local PGNAME. ?PGLNM — AOS/DVS returns a byte pointer to the local PGNAME. ?PGRID — Supply a byte pointer to the PGNAME you want to translate. ?PGRNM — AOS/DVS returns a byte pointer to the PGNAME.
?PGN_AVAIL_SIZE	If offset ?PGN_SUBPKT_IND contains ?PGLID or ?PGRID, then this offset is not used; set it to 0. Otherwise, this offset contains the maximum length of the buffer that contains the process group name.
?PGN_ACT_LEN	If offset ?PGN_SUBPKT_IND contains ?PGLID or ?PGRID, then this offset is not used; set it to 0. Otherwise, AOS/DVS returns the length of the process group name buffer that contained the actual name.

?PIDS

Gets information about PIDs.

?PIDS *[packet address]*
error return
normal return

Input

AC0	One of the following: • Host ID • −1 for the current host • Byte pointer to the host name
AC1	Defines contents of AC0 as follows: • 0 if AC0 contains a host ID • −1 if AC0 contains −1 • Ignored if AC0 contains a byte pointer
AC2	Address of the ?PIDS packet, unless you specify the address as an argument to ?PIDS

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?PIDS packet

Error Codes in AC0

ERHID	Illegal host ID in AC0
ERHNE	Host does not exist
ERPRE	(AC1 = 0 and AC0 <> host ID) or (AC1 = 0 and AC0 <> −1) or (AC1 = −1 and AC0 doesn't contain a byte pointer)
ERPVS	Packet version not supported
ERVBP	Invalid byte pointer passed as system call argument
ERVWP	Invalid address passed as system call argument

Why Use It?

Use this system call to obtain information about the number of processes on your system.

What It Does

You can issue ?PIDS to obtain the maximum number of PIDs allowed. ?PIDS also returns the number of processes currently running on the system and the number of processes below 256.

The process count limit might change in future revisions of the operating system. Furthermore, the number of processes allowed on the system is limited by a system generation question. Programs that allocate resources based on the maximum number of PIDs on a system need a way to obtain this limit; ?PIDS provides the way.

Licensed Material - Property of Data General Corporation

Figure 2–166 shows the structure of ?PIDS's parameter packet, and Table 2–144 describes its contents.

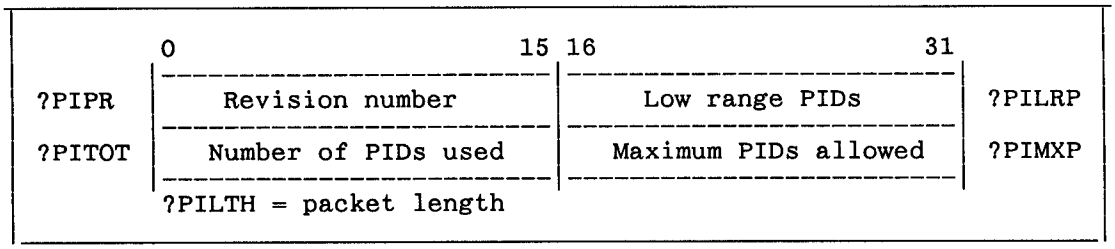


Figure 2–166. Structure of ?PIDS Packet

Table 2–144. Contents of ?PIDS Packet

Offset	Contents
?PIPR	Packet revision number. Place 0 here.
?PILRP	The operating system returns the number of PIDs used that are fewer than 256 (i.e., those in the low range).
?PITOT	The operating system returns the total number of PIDS that are in use. This includes the number of low-range PIDs that are in offset ?PILRP.
?PIMXP	The operating system returns the maximum number of PIDs allowed on the system. This value may differ from that of ?VSPIDS as defined in the parameter files.

?PMTPF

Permits access to a protected file.

?PMTPF
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Packet address

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERAPU	Attempt to pass unheld access privileges
ERCBK	Connection has been broken
ERCDE	Connection does not exist
ERIFI	Invalid protected file ID
ERPRH	Process not in hierarchy
ERRNI	Ring number invalid

Why Use It?

The first opener of a file (?SOPPF) uses ?PMTPF to allow other segment images to open a protected shared file or to revoke this access. ?PMTPF provides you with a way to control access to protected shared files.

What It Does

The first opener of a protected shared file uses ?PMTPF to grant access privileges to other segment images that wish to open the protected shared file. Only the first opener of a protected shared file can issue ?PMTPF against that file.

The ?PMTPF caller cannot grant any access privileges that it does not have itself. Also, ?PMTPF privileges are not cumulative. That is, the ?PMTPF caller grants a segment image only the access privileges that were specified in the last ?PMTPF call that was directed towards that segment image.

Licensed Material - Property of Data General Corporation

The first opener can revoke access privileges by issuing ?PMTPF with a null privileges mask.

A ?PMTPF access grant remains active until one of the following events occurs:

- The connection between the first opener and the target image is broken.
- The first opener closes the file.
- The first opener revokes the access grant.

NOTE: Revoking a ?PMTPF access grant affects only future opens.

To use ?PMTPF, you must supply a packet that contains various information about the target program. Figure 2-167 shows the structure of the packet and Table 2-145 describes its contents.

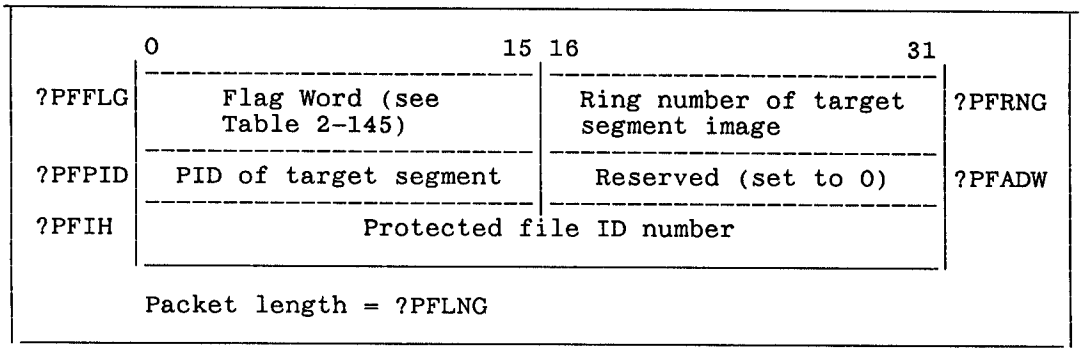


Figure 2-167. Structure of ?PMTPF Packet

Table 2-145. Contents of ?PMTPF Packet*

Offset	Contents
?PFFLG	Flag word ?FAC<O,W,A,R,E>--Access privileges. ?PFFO--First open. ?PFRW--Open for read and write.
?PFRNG	Ring number of target segment.
?PFPID	PID of target segment.
?PFADW	Reserved. (Set to 0.)
?PFIH (double word)	Protected file ID number.

* There is no default unless otherwise specified.

?PNAME

Gets a full process name.

?PNAME
error return
normal return

Input

AC0 Byte pointer to one of the following:

- Buffer to receive the process name
- Name of the target process, to obtain its PID

AC1 One of the following:

- PID of the target process, to obtain its process name
- 0 to obtain the PID of the target process
- −1 to obtain the PID of the calling process

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Requested PID

AC2 Undefined

Error Codes in AC0

ERPRE	Invalid or illegal system call parameter
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?PNAME to obtain both the process name and PID of a process. Therefore, ?PNAME is a useful complement to system calls that require these identifiers as input.

Licensed Material - Property of Data General Corporation

What It Does

?PNAME returns either the PID or the process name of a target process, depending on your input to AC0 and AC1.

To obtain the name of a process other than the calling process, load AC1 with the PID of that process, and load AC0 with a byte pointer to a buffer that will receive the process name. To obtain the PID of a process (except the calling process), load AC1 with 0, and load AC0 with a byte pointer to the process's name. The operating system returns the PID you requested to AC1.

To obtain the PID of the calling process, load AC1 with -1 and AC0 with 0. The operating system returns the caller's PID to the same accumulator. To obtain the name of the calling process, load AC0 with a byte pointer to a receive buffer.

?PRCNX

Passes a connection from one server to another.

?PRCNX
error return
normal return

Input

AC0	Customer's PID
AC1	PID of the new server
AC2	A field that describes the following:

- Bits 0 through 20 are reserved (Set to 0.)
- Bits 21 through 23 contain the ring of the customer
- Bits 24 through 28 are reserved (Set to 0.)
- Bits 29 through 31 contain the ring of the specified server

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection does not exist
ERCCS	Cannot connect to self
ERNAS	Process is not a server
ERPRH	Attempt to access process not in hierarchy
ERRNI	Invalid ring number

Why Use It?

?PRCNX allows you to break a connection between a ring of a server and one of its customers and to re-establish the connection with a ring of a new server process. ?PRCNX is useful for passing a customer/server connection from an intermediate server to a specialized server.

Licensed Material - Property of Data General Corporation

What It Does

?PRCNX passes the connection established between the calling segment image (PID/ring tandem) and one of its customers to a segment image of another server process. Before you issue ?PRCNX, load AC0 with the PID of the customer process, and load AC1 with the PID of the new server. Note that AC1 cannot contain the PID of the calling process.

The calling process must be a server of the customer that you specify in AC0 and AC2, and there must be an unbroken connection between the two. The new server process must have already issued ?SERVE to declare itself a server.

When a server issues ?PRCNX, the operating system updates the connection-table entry so that it contains the PID/ring of the new server and deletes any previous connection between the customer and the new server. In effect, this breaks the old connection. Therefore, the operating system revokes any protected shared file access grants that the old server allowed the customer.

Notes

- See the description of ?PCNX in this chapter.

?PRDB/?PWRB — Performs physical block I/O.

?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

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 2–148.) 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.

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 2-168 shows the structure and Table 2-146 describes the contents of the ?PRDB/?PRWB packet. Table 2-147 lists the disk and tape types applicable to the offsets in the packet extension.

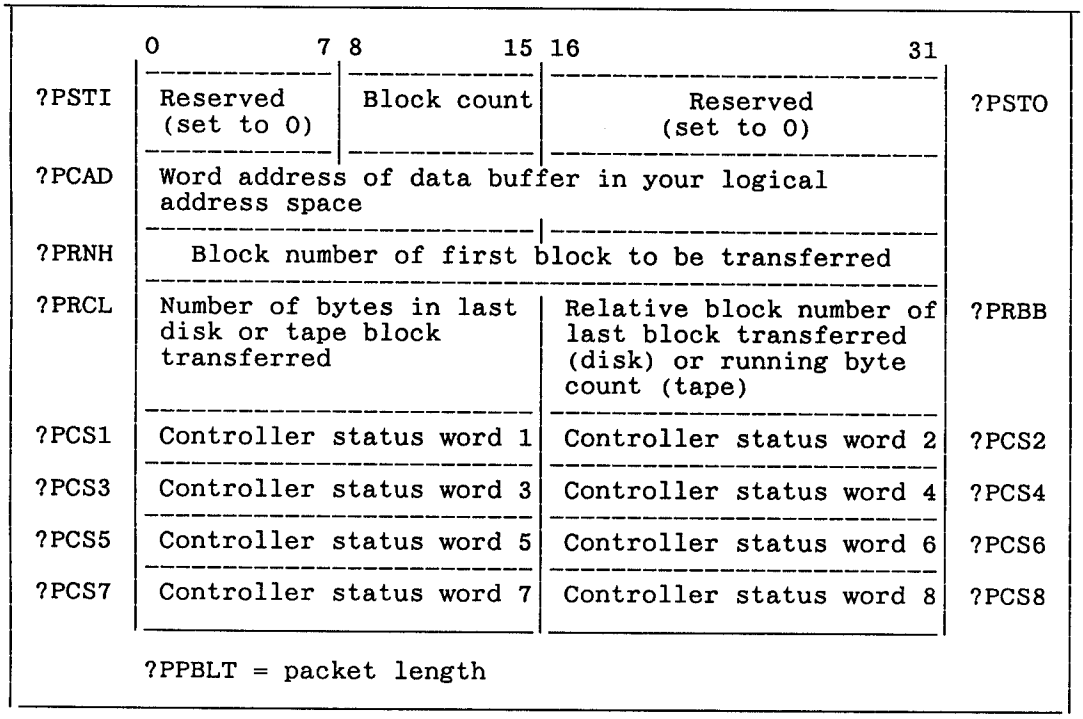


Figure 2-168. Structure of ?PRDB/?PWRB Packet

?PRDB/?PRWB Packet Offsets

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).

?PRDB/?PWRB Continued

As Table 2-146 and Table 2-147 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 2-146. Contents of ?PRDB/?PWRB Packet

Offset	Input Value	Output Value
?PSTI	Left Byte: Reserved. Right Byte: Block count.	Unchanged. Unchanged.
?PSTO	Reserved.	Unchanged.
?PCAD (double word)	Address of data buffer in your logical address space.	Unchanged.
?PRNH (double word)	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 2-147.)

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.

Licensed Material - Property of Data General Corporation

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.

Table 2-147. ?PRDB/?PWRB Packet: Controller Status Words

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

(Continues)

Table 2-147. ?PRDB/?PWRB Packet: Controller Status Words

MTB/MTD Tape	Status Word	Use
6026 (800/1600 bpi) 4307 (1600/6250 bpi) 6200 (1600/6250 bpi) 6300 (1600/6250 bpi)	?PCS1 ?PCS2	DIA (Transfer) Status DIC (Drive) Status
MTC Tape	**Status Word**	**Use**
6125 (1600 bpi) 6231/6311 (Cartridge)	?PCS1	DIA (Transfer) Status

(Concluded)

Table 2-148. Error Reports Returned in ?PRDB/?PWRB Offsets

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

Notes

- Refer to the *Principles of Operation 32-Bit ECLIPSE Systems* manual and the *Programmer's Reference Series - Peripherals* manual for details on the DIA, DIB, and DIC I/O instructions. Or, refer to the programmer's reference manual for your particular model.
- Refer to the *Programmer's Reference Peripherals* manual for the disk-drive error-correcting codes.

?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.

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.

?PRIPR

Changes the priority of a process.

?PRIPR
error return
normal return

Input

AC0 One of the following:

- -1 to change the priority of the calling process
- Byte pointer to the name of the target process
- PID of the target process

AC1 One of the following:

- -1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID

Otherwise, ignores contents of AC1.

AC2 New priority of the target process in the following ranges:

- 1 through 255 for resident and pre-emptible processes
- 1 through 3 for swappable processes

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Licensed Material - Property of Data General Corporation

Error Codes in AC0

ERMPR	System call parameter address error
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERPRP	Illegal priority (Either a caller with privilege ?PVPR tried to raise its own priority or the newly assigned priority is out of range.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?PRIPR changes a process's priority. Process priority is one of several criteria the operating system uses to determine the scheduling order for individual processes.

You can use ?PRIPR to favor a process for CPU access (by raising its priority) or to favor other processes over the target process (by lowering the target's priority).

What It Does

?PRIPR changes the priority of the process that you specify in AC0 (the calling process or another target process). If the caller is in Superprocess mode, it can use ?PRIPR to change the priority of any process. Otherwise, its calling is restricted to itself.

The priority range for all processes is 1 (the highest priority) through 511 (the lowest priority).

Notes

- See the description of ?CTYPE in this chapter.

?PRKIL — Kills all tasks of a specified priority.

?PRKIL
error return
normal return

Input

AC0	Priority level of the tasks to kill
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?

?PRKIL allows you to kill all tasks of a given priority level. (You can group tasks of similar functions under the same priority.)

What It Does

?PRKIL kills (terminates) all tasks of the priority level that you specify in AC0. If the target tasks were explicitly suspended by ?SUS, ?IDSUS, or ?PRSUS, or because they issued an ?XMTW or ?REC, the operating system lifts the suspension and readies the target tasks at the highest priority level (0) before it executes the task-kill logic. If the target tasks became suspended by issuing other system calls, the operating system aborts those outstanding system calls and readies the tasks before it performs the task-kill logic.

If you supplied a kill-processing routine for the target tasks, the operating system passes control to those routines when you issue ?PRKIL. You should end each kill-processing routine with a ?KILL. ?KILL invokes a ?UKIL task termination routine — either the system default ?UKIL or a ?UKIL you have defined. ?UKIL, and then terminates the tasks.

If you did not supply kill-processing routines, control passes to the appropriate ?UKIL routine immediately.

Notes

- See the descriptions of ?IDSUS, ?PRSUS, ?XMTW, ?REC, and ?KILL in this chapter.

?PROC

Creates a process.

?PROC *[packet address]*
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	None
Error Codes	None
Parameter Packet	Some

Input

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

Output

AC0	Undefined
AC1	PID of the new process
AC2	Address of the ?PROC packet

Error Codes in AC0

ERBMX	Illegal maximum process size on process create
ERCON	Console device specification error
ERDAD	Directory access denied
ERDPT	Different process type
EREAD	Execute access denied
EREXC	Resident process tried to ?PROC and block
ERIFT	Illegal file type
ERPDF	Error in process UST definition
ERPRN	Number of processes exceeds maximum
ERPRP	Illegal priority (The caller tried to assign the new process a higher priority than its own, without the privilege to do so.)
ERPRV	Caller not privileged for this action
ERPTY	Illegal process type (The caller tried to assign the new process a process type different from its own, without the privilege to do so.)
ERSMX	Error in setting maximum CPU time
ERSNM	Attempt to exceed maximum number of sons
ERUNM	Illegal username
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument
ERWSM	Attempt to set working set minimum, not privileged
ERWSS	Invalid working set maximum/minimum

?PROC Continued

Why Use It?

?PROC allows you to create a process and define its privileges and characteristics.

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 2–169 shows the structure of the ?PROC packet and Table 2–149 defines each offset and mask.

Among other characteristics, ?PROC defines the process's type and priority, its maximum and minimum working set size, its creation privileges (that is, the right to create sons with additional ?PROC system calls), and its access rights to files. 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 packet lets you implement the following when you issue ?PROC:

- User locality.
- CPU selection.
- ?PROC–related username.
- Unique PIDs.

Figure 2–170 shows the structure of the ?PROC extension packet, and Table 2–151 defines each offset.

Licensed Material – Property of Data General Corporation

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.

- Default access control.

  Bit ?PFDA, the access privileges bit, prevents the operating system from passing the father's file access privileges to the son. 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.

- 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.

Table 2-150 describes each of the privilege bits in offset ?PPRV.

?PROC Continued

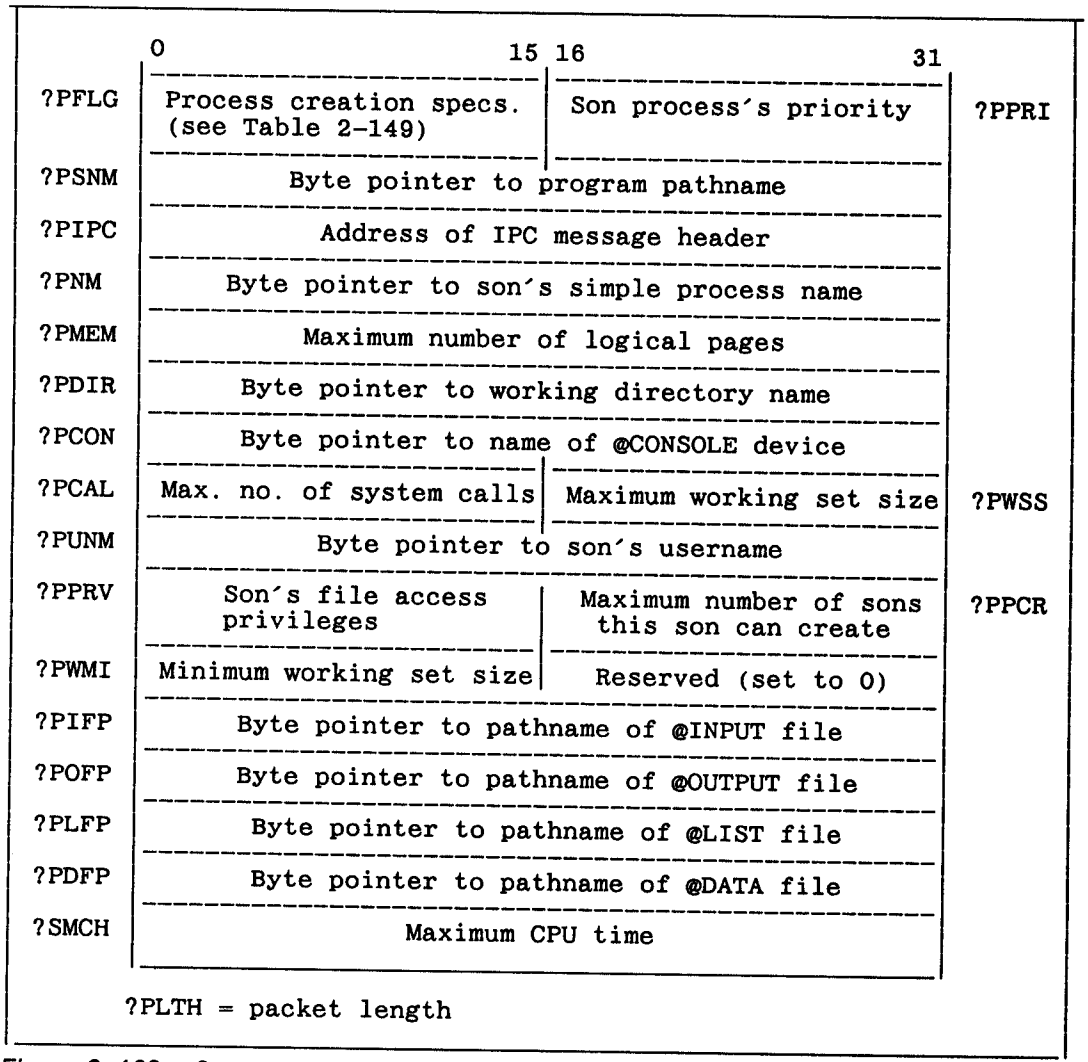


Figure 2-169. Structure of ?PROC Packet

Licensed Material - Property of Data General Corporation

Table 2-149. Contents of ?PROC Packet*

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

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

?PROC Continued

Table 2-149. Contents of ?PROC Packet*

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

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

Table 2-149. Contents of ?PROC Packet*

Offset	Contents
?PCON (double word)	Byte pointer to name of new process's @CONSOLE device; if 0, is associated with @CONSOLE. 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.)
?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 (double word)	Byte pointer to son's username. DEFAULT = -1 (same username as the caller's).
?PPRV	Son's file access privileges; each bit in this offset assigns a privilege if ?PFPM is not set, or denies a privilege if ?PFPM is set. ?PFPM is a bit pattern 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 2-149. 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 (double word)	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 (double word)	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 (double word)	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 (double word)	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 (double word)	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)

Licensed Material - Property of Data General Corporation

Table 2-150 describes each of the privilege bits in offset ?PPRV.

Table 2-150. 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.)
?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-like command line as the IPC message. This way, the son process can access arguments and switch values via a ?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 2-75 for the structure of the ?ISEND header.

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.

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 2-75 and Table 2-51 describe the IPC message header; Figure 2-74 describes offset ?IUFL.

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.

?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.

Licensed Material - Property of Data General Corporation

For example, 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 2-75 and Table 2-51 describe the IPC message header; Figure 2-74 describes offset ?IUFL.

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 every other creation parameter in the ?PROC packet.

- 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.

- 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.

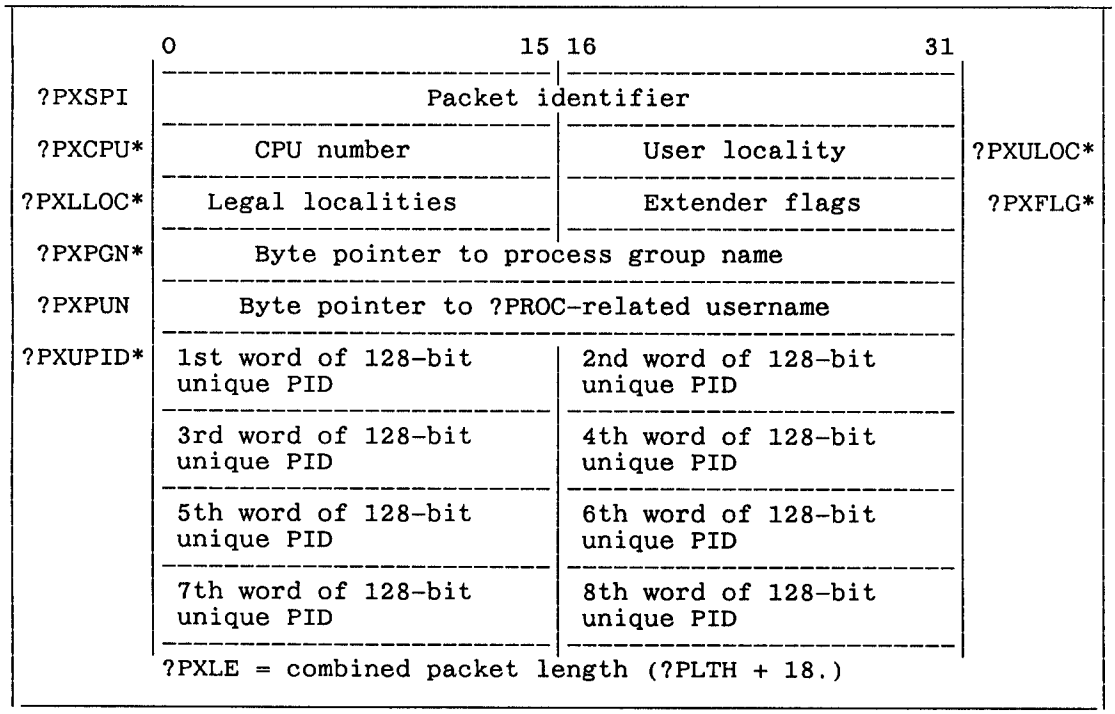
By default, the operating system adjusts the working set parameters dynamically, based on the process's type, page fault history, and general system overhead. When you set ?PWMI and ?PWSS, you direct the operating system to keep the working set within specific limits.

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 the 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. As mentioned before, Figure 2-170 contains the structure of the ?PROC extension packet and Table 2-151 describes its contents.



* This offset differs between AOS/VS and AOS/DVS. See Table 2-151.

Figure 2-170. Structure of ?PROC Extension Packet

In Table 2-151, the column heads "VS" and "DVS" represent "AOS/VS" and "AOS/DVS," respectively. The entries in these columns have the following codes:

---	An operating system does not define a value for the given offset.
U	An operating system handles the offset differently (uniquely).
[no entry]	The operating systems handle the offset in the same way.

Table 2-151. Contents of ?PROC Parameter Packet Extension

Offset	VS	DVS	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/DVS does not use this value.
?PXULOC		---	Place here the user locality as you have defined it. AOS/DVS does not use this value.
?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/DVS ignores this offset.
?PXFLG		---	Extender flags word. This 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 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/DVS does not use this information.
?PXPGN (dbl wd)	---		Byte pointer to the process group name that the new process will be created in. AOS/VS does not use this information.
?PXPUN (dbl wd)			Byte pointer to the ?PROC-related user name that the new process will be created with.
?PXUPID* (8 words)	---		AOS/DVS returns the 128-bit unique PID (UPID) for the new process. AOS/VS does not have unique PIDs, so it does nothing with these words.

Licensed Material - Property of Data General Corporation

Sample Packet

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

        .LOC    PKT+?PFLG       ;Process creation options.
        .WORD   ?PBRK+0         ;Create a break file if process traps
                                ;and son is a swappable process.

        .LOC    PKT+?PPRI       ;Priority of son process
        .WORD   -1              ;is same as caller's priority (-1 is
                                ;the default).

        .LOC    PKT+?PSNM       ;Byte pointer to pathname of program
                                ;file for new process to execute.
        .DWORD  PATH*2          ;Byte pointer to PATH.

        .LOC    PKT+?PIPC       ;Address of IPC message header.
        .DWORD  -1              ;No IPC message header (-1 is the
                                ;default).

        .LOC    PKT+?PNM        ;Byte pointer to new process's simple
                                ;process name.
        .DWORD  -1              ;Son's simple process name is ASCII
                                ;representation of its PID.

        .LOC    PKT+?PMEM       ;Maximum number of log pages in new
                                ;process.
        .DWORD  20.             ;Maximum of 20. log pages in new
                                ;process.

        .LOC    PKT+?PDIR       ;Byte pointer to name of son's
                                ;working directory.
        .DWORD  -1              ;Same initial working directory as
                                ;the caller's (-1 is the default).

        .LOC    PKT+?PCON       ;Byte pointer to name of new
                                ;process's @CONSOLE device.
        .DWORD  0               ;No device is associated with
                                ;@CONSOLE.

        .LOC    PKT+?PCAL       ;Maximum number of system calls son
                                ;can issue at the same time.
        .WORD   3.              ;Son can issue three system calls
                                ;at a time.

        .LOC    PKT+?PWSS       ;Maximum working set size for son.
        .WORD   -1              ;No limit to working set size for
                                ;son.  (The caller must have the
                                ;?PVWS privilege to set this
                                ;parameter.)

        .LOC    PKT+?PUNM       ;Byte pointer to son's username.
        .DWORD  -1              ;Same username as caller's (-1 is
                                ;the default).
```


Licensed Material - Property of Data General Corporation

?PROC Continued

```
.LOC    PKT+?PPRV       ;Son's file access privileges.
.WORD   -1              ;Son has all caller's file access
                        ;privileges.
.LOC    PKT+?PPCR       ;Maximum number of sons this son can
                        ;create.
.WORD   1               ;This son can create one son.
.LOC    PKT+?PWMI       ;Minimum working set size.
.WORD   -1              ;No working set minimum.
.LOC    PKT+?PIFP       ;Byte pointer to pathname of son's
                        ;@INPUT file.
.DWORD  -1              ;Same @INPUT file as caller.
.LOC    PKT+?POFP       ;Byte pointer to pathname of son's
                        ;@OUTPUT file.
.DWORD  -1              ;Same @OUTPUT file as caller.
.LOC    PKT+?PLFP       ;Byte pointer to pathname of son's
                        ;@LIST file.
.DWORD  0               ;There is no @LIST file.
.LOC    PKT+?PDFP       ;Byte pointer to pathname of son's
                        ;@DATA file.
.DWORD  0               ;There is no @DATA file.
.LOC    PKT+?SMCH       ;Maximum CPU time allotted to son.
.DWORD  -1              ;Son receives remainder of father's
                        ;time limit.
.LOC    PKT+?PLTH       ;End of packet.
```


Licensed Material - Property of Data General Corporation

?PROFILE

Performs a profile request.

?PROFILE *[packet address]*
error return
normal return

Operating System Differences

The function of getting the next profile name is available only under AOS/DVS.

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?PROFILE packet

Error Codes in AC0

ERBTL	Source field is too large
EREDX	Profile entry nonexistent
EREOF	End of file
ERIRB	Destination field too small
ERPAI	Profile access has already been initialized
ERPAN	Profile access has not been initialized
ERPFD	Error in profile field descriptor packet
ERPRE	Invalid parameter passed as a system call argument
ERPRV	Caller not privileged for this action
ERPVS	Packet revision not supported
ERRVN	Reserved value not zero
ERUNM	Illegal username
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

Use this system call to obtain information about, create, delete, and change profiles. You can also retrieve the names of all profiles on the system — a function similar to that of ?GNFN.

?PROFILE Continued

What It Does

This system call lets you perform all the functions of utility program PREDITOR (user profile editor) from within your application program. You must have Superuser privilege to issue ?PROFILE. These functions include the following:

- Creating a profile.
- Renaming a profile.
- Deleting a profile.
- Initiating access to a profile.
- Terminating access to a profile.
- Reading a field in a profile.
- Updating a field in a profile.
- Getting the next profile name (AOS/DVS only).

You may perform only one of these functions when you issue ?PROFILE.

Figure 2-171 shows the structure of the ?PROFILE parameter packet, and Table 2-152 describes its contents.

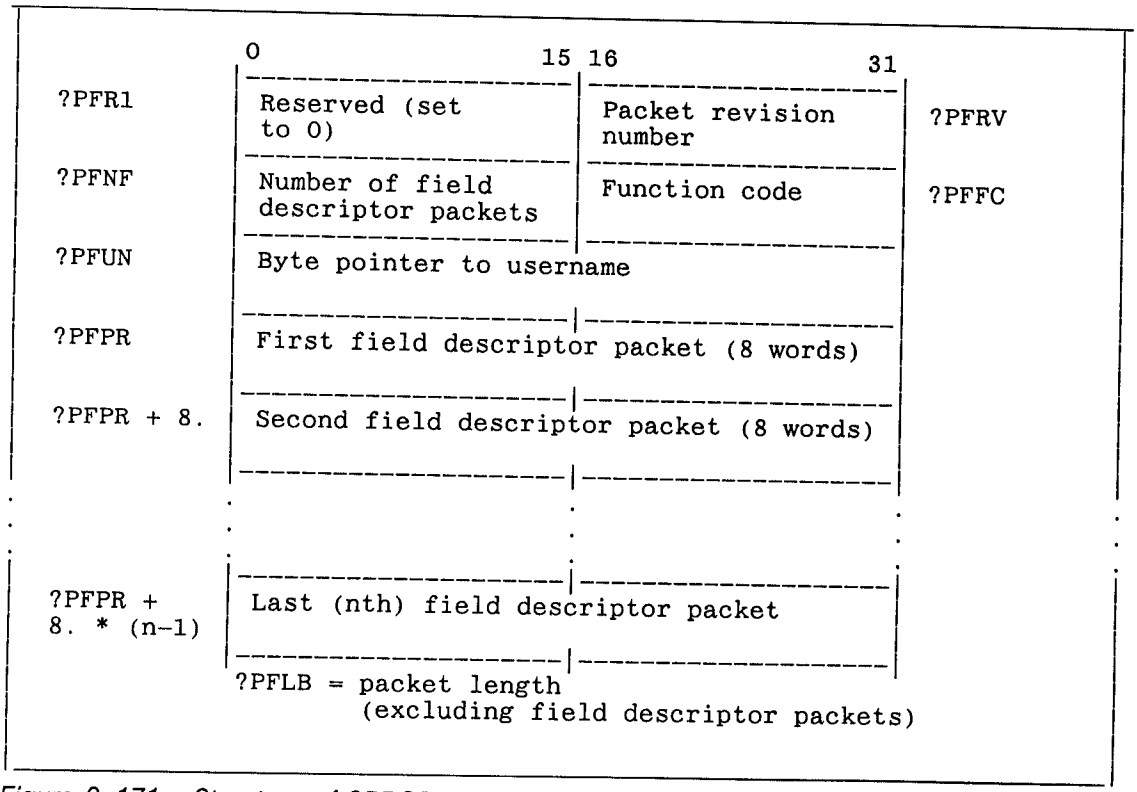


Figure 2-171. Structure of ?PROFILE Parameter Packet

Table 2-152. Contents of ?PROFILE Parameter Packet

	0 15	16 31	
?PFR1	Reserved (Set to 0.)	Current revision of this system call packet. Place ?PKR0 here.	?PFRV
?PFNF	Number of field description packets. They begin at offset ?PFPR. This number is zero for some functions (read a field) and at least one for other functions (create a profile).	Function code for the profile. ?PFCRE - Create ?PFREN - Rename ?PFDEL - Delete ?PFIAC - Initiate access ?PFTAC - Terminate access ?PFRDF - Read field ?PFUFD - Update field ?PFGNP - Get next profile name	?PFFC
?PFUN (double word)	Byte pointer to username. Null terminate this string. This offset has a special use when you select the ?PFGNP function. In this case, place zero here; the operating system will then use this offset for itself.		
?PFPR (8. words)	Field descriptor packet number 1. See Figure 2-172.		
?PFPR + 8. (8. words)	Field descriptor packet number 2. See Figure 2-172.		
. . .	. . .	. . .	. . .
?PFPR + 8. * (n-1)	Field descriptor packet number n. See Figure 2-172.		

Figure 2-172 shows the structure of each field descriptor packet. The packets (if any) begin at offset ?PFPR of the ?PROFILE parameter packet.

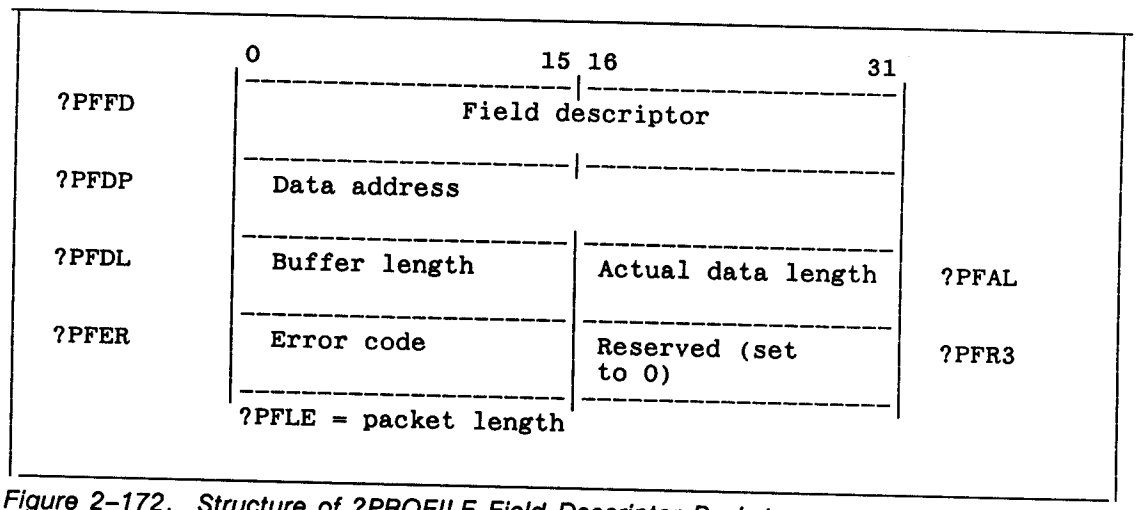
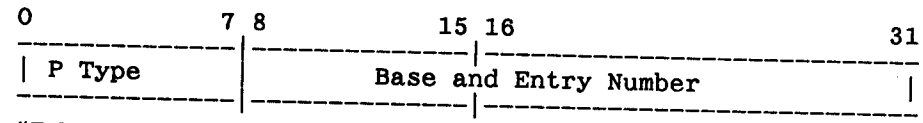


Figure 2-172. Structure of ?PROFILE Field Descriptor Packet

Here are the 8 words of each field descriptor packet. Their descriptions continue over the next several pages.

?PFFD — (double word) field descriptor.

It describes the data item in the profile that you want to retrieve or modify. Its format is



"P Type" means parameter type. Its values and their meanings are

?PFBY — Offsets ?PFDP and ?PFDL are byte oriented.
?PFBI — Offset ?PFDP contains a value; ignore ?PFDL.
?PFWD — Offsets ?PFDP and ?PFDL are word oriented.

The 8 bits of the base usually contain ?PFSE. See PARU.32 for other possible values of these bits.

The rightmost 16 bits of offset ?PFFD (entry number) have the following possible values. Entries with a bit length of 1 bit are set/returned in the most significant bit (1B 0). For example, suppose you want to know if a profile has the Superuser privilege. Set word ?PFFD to ?PFBI!?PFSE!?PSUSER (or ?PFBI+?PFSE+?PSUSER) and the other 7 words of the field descriptor packet to zero. The operating system returns 1S0 in ?PFDP if Superuser privilege is present and 0S0 otherwise.

?PFFD — (double word) field descriptor, continued.

- ?PPASSWD — system password, less than or equal to 16 bytes including a null byte.
- ?PICROG — initial program, less than or equal to 64 bytes including a null byte.
- ?PICCFN — initial interprocessor communications (IPC) filename, less than or equal to 64 bytes including a null byte.
- ?PMXSONS — maximum number of sons, 1 word.
- ?PSOPIO — son's priority, 1 word.
- ?PDISKLM — user's CPD (control point directory) limit, 2 words.
- ?PLOGON — date/time of last logon, 6 words; internal format is

  word 0 — years
  word 1 — months
  word 2 — days
  word 3 — hours
  word 4 — minutes
  word 5 — seconds

- ?PBATCHP — batch priority, 1 word.
- ?PCNSPRV — console usage privilege, 1 bit.
- ?PBCHPRV — batch usage privilege, 1 bit.
- ?PMODPRV — modem usage privilege, 1 bit.
- ?PVCNPRV — virtual console usage privilege, 1 bit.
- ?PRRAPRV — remote resource access privilege, 1 bit.
- ?PPWDPRV — change password privilege, 1 bit.

?PROFILE Continued

?PFFD — (double word) field descriptor, continued.

- ?PNCRYPT — system password encrypted indicator, 1 bit.
- ?PMGSYS — system manager privilege, 1 bit.
- ?PCOMMNT — user comment, less than or equal to 80 bytes including a null byte.
- ?PBWSS — batch working set size, 2 words; the internal format is

  word 0 — minimum
  word 1 — maximum

- ?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 — minimum
  word 1 — maximum

- ?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.

?PFFD — (double word) field descriptor, continued.

- ?PWSON — wide son privilege, 1 bit.

- ?PMCTS — 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 — hard privileges, 1 word; the internal format is

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

- ?PSFTPRV — soft privileges, 1 word; the internal format is

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

- ?PPRCINF — PROC information, 13 words; 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.

- ?PDLOCY — default locality, 1 word.

?PROFILE Continued

?PFFD — (double word) field descriptor, continued.

- ?POLOCY — other localities, 1 word.
- ?PQLOCY — other localities enable, 1 word.
- ?PRCRYPT — remote password encrypted flag, 1 bit.
- ?PRPSSWD — remote password, less than or equal to 16 bytes including a null byte.
- ?PFVER — system profile revision, 2 words.
- ?PN1FLG — flag word, 1 word.
- ?PN2FLG — flag word, 1 word.
- ?PRHRDPR — remote hard privileges, 1 word.
- ?PRSFTPR — remote soft privileges, 1 word.

?PFDP — data address (double word). Enter the byte or word address of the buffer that the operating system returns data in, the byte or word address of the data you want to write, or the data itself that you want to write. You specify the address type (byte or word) by the parameter type portion of offset ?PFFD. For the rename function, offset ?PFDP contains a byte pointer to the null-terminated new username.

?PFDL — buffer length. Enter the length of either the data buffer (for reads) or the length of the data you want to write (for updates). You specify the data size in the units declared by the parameter type portion of offset ?PFFD.

?PFAL — actual data length. After a read, the operating system returns the actual size of the specified field. After an update, the operating system returns the data length specified in offset ?PFDL. You specify the data size in the units declared by the parameter type portion of offset ?PFFD.

?PFER — error code. If the operating system detects an error when processing the field descriptor or when retrieving data that this field descriptor specifies, it returns ERPFD in AC0 and the actual error code in offset ?PFER.

?PFR3 — reserved; set to 0.

?PRRDY Readies all tasks of a specified priority.

?PRRDY
error return
normal return

Input

AC0	Priority number of the tasks to ready
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?

You can use ?PRRDY to ready all suspended tasks of a given priority level to be rescheduled and executed. As we noted earlier, the operating system neither executes nor schedules suspended tasks. Given more than one ready task at the same priority, the operating system schedules and executes them on a round-robin basis. Round robin means that the first task in the priority queue executes first, and so on, down the list. Note that ?PRRDY does not change the relative positions of the tasks within a priority level.

What It Does

?PRRDY readies, that is, lifts from suspension, all tasks of the priority level that you specify in AC0, provided the tasks were suspended with ?SUS, ?IDSUS, or ?PRSUS. If the target tasks were suspended for other reasons, such as outstanding system calls, ?XMT, or ?REC, the operating system waits for the appropriate events to complete before it readies the tasks. If there are no tasks at the specified priority level, the operating system ignores the ?PRRDY and takes the normal return.

When you use ?PRRDY to ready tasks that have lower priorities than the calling task, the operating system reschedules the tasks.

Notes

- See the descriptions of ?SUS, ?IDSUS, ?PRSUS, ?XMT, and ?REC in this chapter.

?PRSUS Suspends all tasks of a specified priority.

?PRSUS
error return
normal return

Input

AC0	Priority level of the tasks to suspend
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?

?PRSUS allows you to suspend more than one task at a time. For example, you might group several tasks together under the same priority level, and then suspend the entire set with ?PRSUS.

What It Does

?PRSUS is the inverse of ?PRRDY; that is, it suspends all tasks of a given priority level. The tasks remain suspended until you ready them by issuing a ?PRRDY against the same priority level, an ?IDRDY against each task, or a ?PRKIL to terminate all tasks at that level.

Before you issue ?PRSUS, load AC0 with the priority level of the tasks to suspend. Note that ?PRSUS also suspends the calling task, if it is a member of the priority group specified in AC0. If there are no tasks at the specified priority level, the operating system ignores the ?PRSUS and takes the normal return.

Notes

- See the description of ?PRRDY in this chapter.

Licensed Material - Property of Data General Corporation

?PSLIST Sets a process group name search list.

AOS/DVS only

?PSLIST
error return
normal return

Input

AC0	Reserved (Must be 0.)
AC1	Reserved (Must be 0.)
AC2	Byte pointer to the new search list

Output

AC0	Undefined
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERSRE	Search list resolution error (There are too many process group names in the search list.)
ERVBP	Invalid byte pointer

Why Use It?

You can access processes outside your own process group by using a search list.

What It Does

?PSLIST sets the process group name search list for the calling process. The symbol ?MXPL represents the maximum length for a process group name search list. Such a search list cannot contain more than eight process group names.

?PSLIST Continued

Use the following format for the process group name search list string:

process-group-name<terminator>[process-group-name<terminator>...]<null>

where

- terminator is either a null <000> or a NEW LINE <012>.

You must conclude the string with a final null, e.g.,

PGROUP1<012>PGROUP2<12><0>

Notes

- See the description of ?SLIST in this chapter.

Licensed Material - Property of Data General Corporation

?PSTAT Returns status information on a process.

?PSTAT *[packet address]*
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	None
Error Codes	None
Parameter Packet	Some

Input

AC0 One of the following:

- PID of the target process
- Byte pointer to the name of the target process
- −1 for status information about the calling process

AC1 One of the following:

- −1 if AC0 is a byte pointer
- Any other value if AC0 contains either −1 or a PID

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

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERIRB	Insufficient room in buffer (for username, ?PROC–related username, memory descriptors, or process group name)
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERPVS	Packet revision not supported
ERVBP	Illegal byte pointer in AC0
ERVWB	Packet failed read/write validation
ERVWP	Illegal word pointer in AC2

Why Use It?

Like ?RUNTM, ?WHIST, and ?XPSTAT, ?PSTAT helps you to determine how well a process competes with others of the same type for system resources, such as memory and CPU time. Note that the operating system updates most of the ?PSTAT information as the process executes.

What It Does

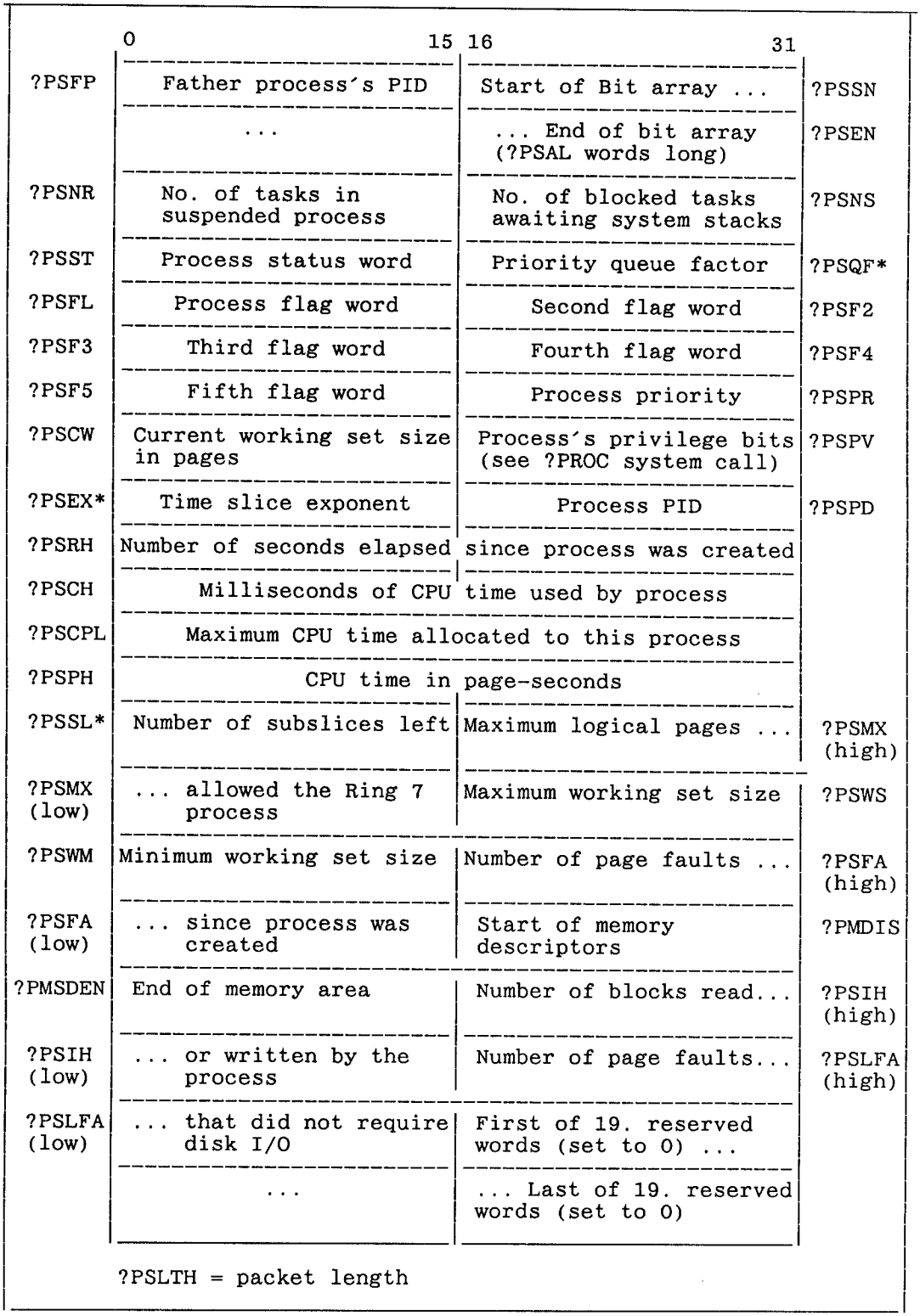
?PSTAT returns internal statistics about the calling process or about any process the caller specifies in AC0. The operating system returns the statistics in the packet. Figure 2–173 shows the structure and Table 2–153 describes the contents of the ?PSTAT packet.

The ?PSAL words in the packet return information about son PIDs from 1 through 255. However, system call ?SONS provides you with the information in these offsets — and with information about son PIDs greater than 255. System call ?XPSTAT has almost all the functionality of ?PSTAT along with much new functionality and considerably less execution time. We recommend that you use ?XPSTAT and/or ?SONS instead of ?PSTAT in new programs.

In addition to the process statistics, the operating system returns seven sets of memory descriptors in the packet, one descriptor for each of the rings 1 through 7. Each descriptor contains information about the program currently running in that ring. Figure 2–174 shows the memory descriptor structure.

Licensed Material - Property of Data General Corporation

Parameter Packet



* This offset differs between AOS/VS and AOS/DVS. See Table 2-153.

Figure 2-173. Structure of ?PSTAT Packet

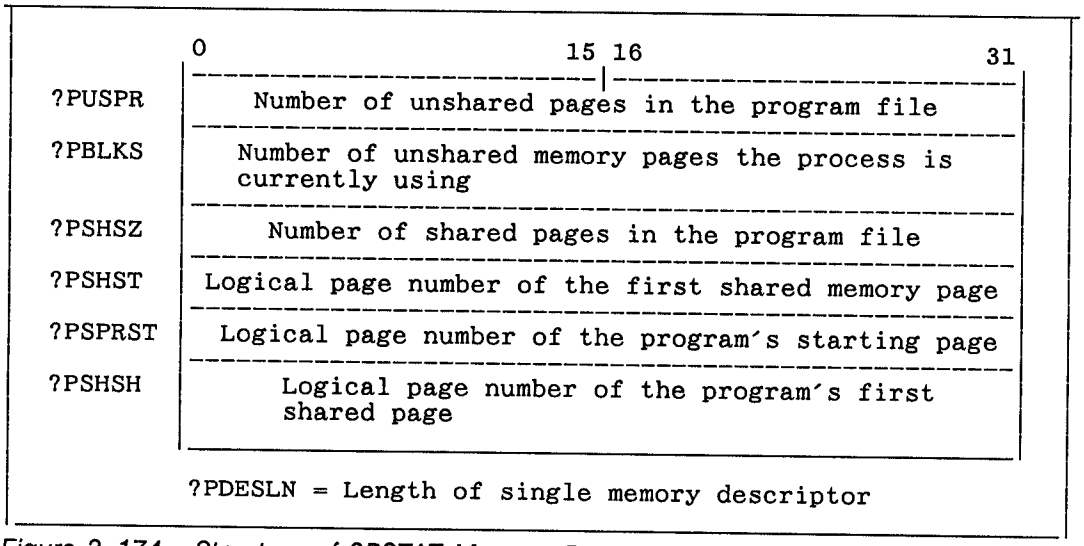


Figure 2-174. Structure of ?PSTAT Memory Descriptor

In Table 2-153, the column heads "VS" and "DVS" represent "AOS/VS" and "AOS/DVS," respectively. The entries in these columns have the following codes:

---	An operating system does not define a value for the given offset.
U	An operating system handles the offset differently (uniquely).
[no entry]	The operating systems handle the offset in the same way.

Table 2-153. Contents of ?PSTAT Parameter Packet

Offset	VS	DVS	Description
?PSFP			PID of the process's parent process. The operating systems reserve a 16-bit word for the small PID. User programs that mask the value returned to 8 bits (length of "low pids") will not return correct PID values for creating processes with hybrid- or anyPIDs.
?PSSN			Bit array that identifies those processes subordinate to the target processes that have small PIDs. (Use the ?SONS system call for PID information on subordinate processes with hybrid- or anyPIDS.) The position of the bits indicates the PID numbers. If Bit 33 is set, for example, the target process has a subordinate process with a PID of 33. This bit array is ?PSAL words long.
?PSEN			End of the bit array.
?PSAL			Bit array length in words.
?PSNR			Number of tasks in the process currently suspended on ?IREC system calls.
?PSNS			Number of tasks that are blocked awaiting system stacks.
?PSST			Process status word (Bit 7: process is blocked).
?PSQF		---	Priority queue factor (indicates the process's priority for execution relative to other processes of the same type). AOS/DVS does not return this information.
?PSFL			Process flag word ?PSSP--process is swappable ?PSPP--process is pre-emptible ?PSTA--type A process. ?PSTB--type B process. ?PSTC--type C process.
?PSF2			Second flag word.

(Continues)

Table 2-153. Contents of ?PSTAT Parameter Packet

Offset	VS	DVS	Description
?PSF3			Third flag word.
?PSF4			Fourth flag word.
?PSF5			Fifth flag word.
?PSPR			Process priority.
?PSCW			Current number of shared and unshared pages in process's logical address space (the OS user-defined "working set" size).
?PSPV			Process's privilege bits. (See ?PROC in this system call dictionary for information on access privileges.)
?PSEX	U	U	Current time slice exponent (value indicating the amount of CPU time currently allocated to the target process). Systems determine value heuristically using "subslice," a constant, and "s", the time-slice exponent: VS: (time slice exponent range of 1 - 6) time slice = subslice * 2**s) AOS/DVS: (time slice exponent range of X - X) time slice = subslice * XXX (See offset ?PSSL in this table and the functional description below.)
?PSPD			PID of the process.
?PSRH (dbl wd)			Number of seconds that have elapsed since the process was created.
?PSCH (dbl wd)			CPU time (in milliseconds) used by the process.
?PSCPL (dbl wd)			Maximum CPU time allocated to this process.
?PSPH (dbl wd)			CPU time (in page-seconds).

(Continued)

Table 2-153. Contents of ?PSTAT Parameter Packet

Offset	VS	DVS	Description
?PSSL	U	U	Number of remaining subslices in the process's time slice. This value is based on the current time slice exponent returned in ?PSEX. If the time slice exponent is 1, an AOS/VS process has two remaining subslices (that is, 2**1). AOS/DVS does not break time slices into subslices.
?PSMX (dbl wd)			Maximum number of logical pages allowed the process in Ring 7.
?PSWS			Maximum working set size in pages.
?PSWM			Minimum working set size in pages.
?PSFA (dbl wd)			Number of page faults since the process was created (with ?PROC).
?PMDIS			Start of memory descriptors.
?PMSDEN			End of memory area.
?PSIH (dbl wd)			Number of blocks read or written by the process.
?PSLFA (dbl wd)			Number of page faults that did not require disk input/output.

(Concluded)

Notes

- See the descriptions of ?RUNTM and ?WHIST in this chapter.
- The parameters shown in Figure 2-174 are offsets from the memory descriptor base address. To get the base address for a particular ring, use the following formula:

  <base_of_packet> + ?PMDIS + (?PDESLN * (ring number-1))

  where

  - ?PMDIS is the starting offset in the ?PSTAT packet that contains the memory descriptors.
  - ?PDESLN is the length of each memory descriptor.

  Assume, for example, that you loaded a program file into Ring 5 with the ?RINGLD system call. To obtain the base address of the descriptor for that ring, you would use the following formula:

  <base_of_packet> + ?PMDIS + (?PDESLN * 4)

Sample Packet

```
PKT:  .BLK    ?PSLTH          ;Allocate enough space for packet.
                              ;Packet length = ?PSLTH.
      .LOC    PKT+?PSFP       ;PID of father process.
      .WORD   0
      .LOC    PKT+?PSSN       ;Bit array showing PIDs of process's
      .WORD   0               ;sons.
      .LOC    PKT+?PSEN       ;End of bit array.
      .WORD   ?PSAL
      .LOC    PKT+?PSNR       ;Number of tasks in the process that
      .WORD   0               ;are currently suspended on ?IREC
                              ;system calls.
      .LOC    PKT+?PSNS       ;Number of tasks that are blocked
      .WORD   0               ;awaiting system stacks.
      .LOC    PKT+?PSST       ;Process status word.
      .WORD   0
      .LOC    PKT+?PSQF       ;Priority queue factor.
      .WORD   0
      .LOC    PKT+?PSFL       ;Process flag word.
      .WORD   0
      .LOC    PKT+?PSF2       ;Second flag word.
      .WORD   0
      .LOC    PKT+?PSF3       ;Third flag word.
      .WORD   0
      .LOC    PKT+?PSF4       ;Fourth flag word.
      .WORD   0
      .LOC    PKT+?PSF5       ;Fifth flag word. (Reserved; set
      .WORD   0               ;to 0.)
      .LOC    PKT+?PSPR       ;Process priority.
      .WORD   0
```

```
.LOC    PKT+?PSCW       ;Number of pages in current working
.WORD   0               ;set.
.LOC    PKT+?PSPV       ;Process's privilege bits.
.WORD   0
.LOC    PKT+?PSEX       ;Current time slice exponent for the
.WORD   0               ;target process.
.LOC    PKT+?PSPD       ;PID of the process.
.WORD   0
.LOC    PKT+?PSRH       ;Number of seconds elapsed since
.DWORD  0               ;process creation.
.LOC    PKT+?PSCH       ;CPU time process used (in
.DWORD  0               ;milliseconds).
.LOC    PKT+?PSCPL      ;Maximum CPU time allocated to this
.DWORD  0               ;process.
.LOC    PKT+?PSPH       ;Page usage over CPU time (in pages/
.DWORD  0               ;second).
.LOC    PKT+?PSSL       ;Subslices remaining in process's
.WORD   0               ;time slice.
.LOC    PKT+?PSMX       ;Maximum number of logical pages
.DWORD  0               ;allowed this process in Ring 7.
.LOC    PKT+?PSWS       ;Maximum working set size (in pages).
.WORD   0
.LOC    PKT+?PSWM       ;Minimum working set size (in pages).
.WORD   0
.LOC    PKT+?PSFA       ;Number of physical page faults since
.DWORD  0               ;process creation.
.LOC    PKT+?PMDIS      ;Start of memory descriptors.
.WORD   0
.LOC    PKT+?PMSDEN     ;End of memory area.
.WORD   0
.LOC    PKT+?PSIH       ;Number of blocks read or written by
.DWORD  0               ;process.
.LOC    PKT+?PSLFA      ;Number of logical page faults.
.DWORD  0
.LOC    PKT+?PSLTH      ;End of packet.
```

?PTRDEVICE Controls input from a pointer device.

?PTRDEVICE *[packet address]*
error return
normal return

Input

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

Output

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

Error Codes Returned in AC0

ERIPP	Invalid pointer device parameter
ERPKT	Invalid packet ID
ERRVN	A reserved value not 0
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid address passed as a system call argument
ERWNE	The window you specified does not exist

Error codes from the file system

Why Use It?

?PTRDEVICE lets you control the type of input you receive from a pointer device. Input from a pointer device can tell you what the user is currently doing with the pointer device (such as moving the pointer into a window or pressing a pointer device button). You can also use ?PTRDEVICE to find out the current status and location of the pointer.

?PTRDEVICE is useful only if your program will run in an AOS/VS or AOS/DVS windowing environment. The call applies to a single window at a time. For example, if you want to find out the status of the pointer for every window belonging to your program, you must issue the ?PTRDEVICE call repeatedly, specifying a different window each time.

?PTRDEVICE is a multifunctional system call. To perform a particular function, you set the offset ?PTRDEV_PKT.FUNC (in the main ?PTRDEVICE packet) to the function code you want. Table 2-154 lists the valid function codes and the functions they perform.

Table 2-154. ?PTRDEVICE Function Codes

Function Code	What It Lets You Do
?PTRDEV_SET_EVENTS	Specify the types of pointer events you want to receive in a window's input buffer. (See the section "Selecting the Pointer Events You Want.")
?PTRDEV_SET_DELTA	Specify how far the pointer device should move in a window before you receive a movement event. (See the section "Specifying a Pointer Delta.")
?PTRDEV_LAST_EVENT	Return information about the most recent pointer event to occur in a window. (See the section "Getting Information About the Last Pointer Event.")
?PTRDEV_SET_POINTER	Control the appearance of the pointer within a window (enable or disable the pointer, and/or set its shape). (See the section "Controlling the Appearance of the Pointer.")
?PTRDEV_GET_PTR_STATUS	Return the current pointer device settings for a window (which events you have requested, the current shape of the pointer, whether or not it is visible, and so on.) (See the section "Getting the Pointer Status.")
?PTRDEV_GENERATE_EVENT	Generate a pointer event as if it had come from the physical pointer device. Useful for moving the pointer or simulating pointer device input. (See the section "Moving the Pointer.")

The Main ?PTRDEVICE Packet

When you issue a ?PTRDEVICE call, you set up both a main packet (common to all functions of the call) and a subpacket (unique for each function). The structure for the main packet appears in Figure 2-175; we cover the subpackets later, when we discuss each function.

?PTRDEVICE Continued

The ?PTRDEVICE call applies to one window at a time; in the main packet, you specify which window you want. You can specify a window in one of three ways:

- Channel number — When you open an I/O channel to a window (using ?OPEN), the operating system returns a channel number.
- Window pathname — When you create a window, you give it a window name. A window's pathname takes the form @PMAPn:windowname, where @PMAPn is the name of the pixel-mapped console that contains the window. The window pathname must end in a null character, <0>; it can be a maximum of ?MXPL characters long (including the null terminator).
- Window ID number — When you create a window, the operating system returns the window ID number in the main ?WINDOW packet.

Set the offset ?PTRDEV_PKT.FLAGS to indicate which method you are using to specify a target window (see Table 2-155). You can use only one method at a time. Fill in the appropriate field in the packet (channel number, pathname, or ID number), and set the other two fields to zero.

In addition to specifying the target window, you supply the following information in the main ?PTRDEVICE packet:

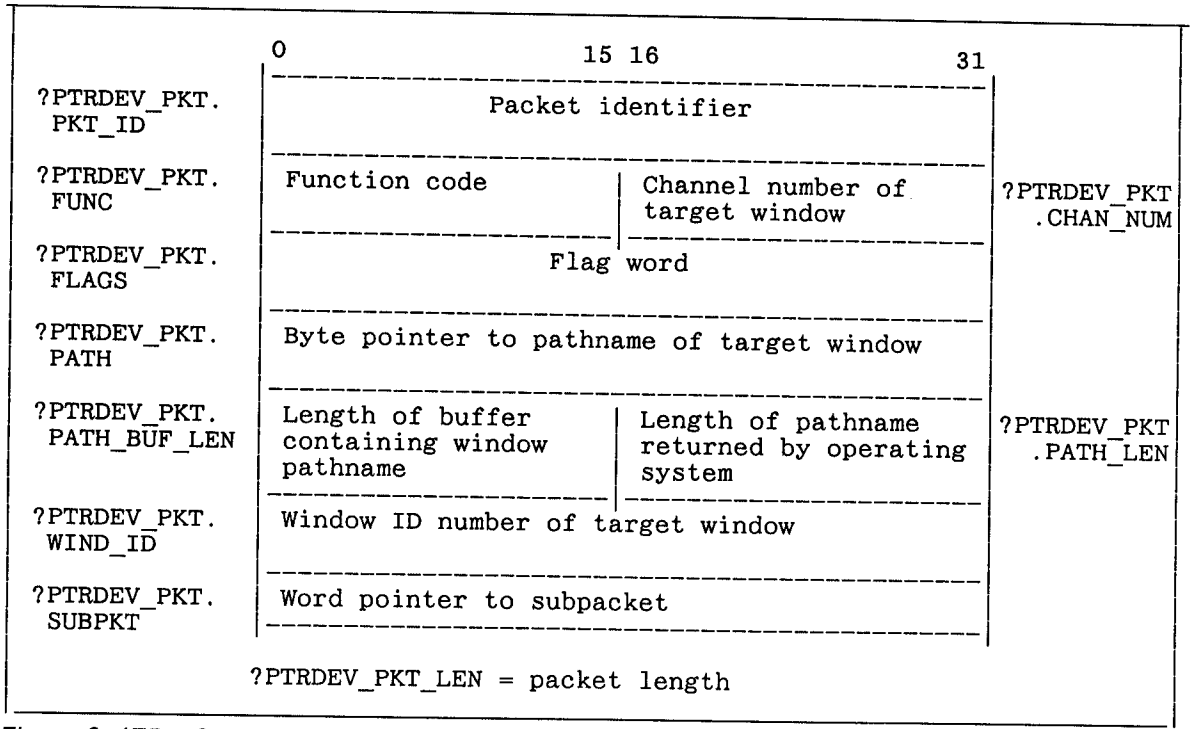


Figure 2-175. Structure of the ?PTRDEVICE Main Packet

Table 2-155. Contents of the ?PTRDEVICE Main Packet

Offset	Contents
?PTRDEV_PKT.PKT_ID (double word)	Packet identifier; set to ?PTRDEV_PKT_PKTID.
?PTRDEV_PKT.FUNC	The code for the function you want. You must always supply a function code. (Function codes are listed in Table 2-154.)
?PTRDEV_PKT.CHAN_NUM	Channel number of target window. When specifying the window by its channel number, you must also set flag ?PTRDEV_PKT.FLAGS.IN_CHAN in flag word ?PTRDEV_PKT.FLAGS. Set this offset to 0 when specifying the target window by its pathname or window ID.
?PTRDEV_PKT.FLAGS (double word)	Flag word ?PTRDEV_PKT.FLAGS.IN_PATH -- Set this flag when specifying the target window by its pathname. ?PTRDEV_PKT.FLAGS.IN_WIND_ID -- Set this flag when specifying the target window by its window ID. ?PTRDEV_PKT.FLAGS.IN_CHAN -- Set this flag when specifying the target window by its channel number.
?PTRDEV_PKT.PATH (double word)	Byte pointer to pathname of target window. When specifying the window by its pathname, you must also set flag ?PTRDEV_PKT.FLAGS.IN_PATH in flag word ?PTRDEV_PKT.FLAGS. Set this offset to 0 when specifying the target window by its channel number or window ID.

(Continues)

Table 2-155. Contents of the ?PTRDEVICE Main Packet

Offset	Contents
?PTRDEV_PKT.PATH_BUF_LEN	Length of the buffer containing the window pathname. The buffer length must include the null terminator; the maximum buffer length is ?MXPL.
?PTRDEV_PKT.PATH_LEN	Set to 0. For functions that return a pathname, the length of the pathname is returned here.
?PTRDEV_PKT.WIND_ID (double word)	Window ID number of target window. When specifying the window by its window ID, you must also set flag ?PTRDEV_PKT.FLAGS.IN_WIND_ID in flag word ?PTRDEV_PKT.FLAGS. Set this offset to 0 when specifying the target window by its pathname or channel number.
?PTRDEV_PKT.SUBPKT (double word)	Word pointer to the subpacket for the function. If the function does not require a subpacket, set this offset to 0.

(Concluded)

Selecting the Pointer Events You Want

A pointer event is a 16-byte sequence that describes an action the user has just taken with the pointer device. The operating system inserts pointer events into the input data stream of the window in which the event took place. From the input stream, the events go into the window's input buffer, where they stay until you issue a ?READ on that window.

You must tell the operating system what types of pointer events you want it to send to each window. By default, each window receives no pointer events; to receive events, issue the ?PTRDEVICE call with function code ?PTRDEV_SET_EVENTS. In the main ?PTRDEVICE packet, specify the window for which you are setting the events, and provide a word pointer to the ?PTRDEV_SET_EVENTS subpacket. Figure 2-176 shows the subpacket structure.

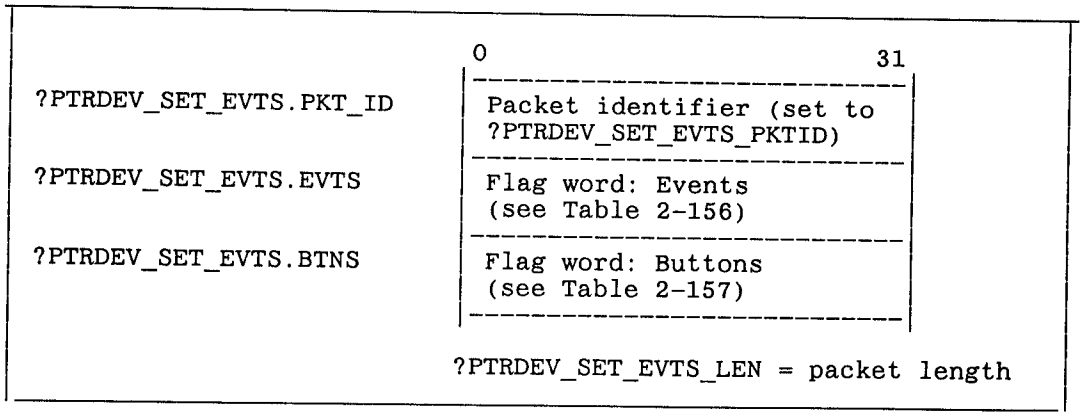


Figure 2-176. Structure of the ?PTRDEV_SET_EVENTS Subpacket

The ?PTRDEV_SET_EVENTS subpacket contains flag words that let you specify

- The pointer events you want to receive. You can specify one or more pointer events. (See Table 2-156.)
- The buttons in which you are interested (if you request pointer events that concern buttons). (See Table 2-157.)

Table 2-156. Flags for Selecting Pointer Events (Flag Word ?PTRDEV_SET_EVTS.EVTS)

Flag	Description
?PTRDEV_SET_EVTS.EVTS.MOVEMENT	Set this flag to receive movement events that occur in this window. (A movement event occurs if the pointer moves while it is within the window.)
?PTRDEV_SET_EVTS.EVTS.WINDOW	Set this flag to receive window events that occur in this window. There are four kinds of window events: Enter window Exit window Activate window Deactivate window
?PTRDEV_SET_EVTS.EVTS.BTN_DOWN	Set this flag to receive button down events for this window. (A button down event occurs if the user presses a button while the pointer is within the window.)
?PTRDEV_SET_EVTS.EVTS.BTN_UP	Set this flag to receive button up events for this window. (A button up event occurs if the user releases a button while the pointer is within the window.)
?PTRDEV_SET_EVTS.EVTS.DBL_CLICK	Set this flag to receive double click events for this window. (A double click event occurs when the user presses and releases a button twice quickly.)

Table 2-157. Flags for Selecting Buttons (Flag Word ?PTRDEV_SET_EVTS.BTNS)

Flag	Description
?PTRDEV_SET_EVTS.BTNS.ONE	Set this flag to receive button events that concern pointer device button 1.
?PTRDEV_SET_EVTS.BTNS.TWO	Set this flag to receive button events that concern pointer device button 2.
?PTRDEV_SET_EVTS.BTNS.THREE	Set this flag to receive button events that concern pointer device button 3.

Specifying a Pointer Delta

The pointer delta is the number of pixels the pointer must move before the operating system sends you a movement event. To specify the pointer delta, issue the ?PTRDEVICE call, function code ?PTRDEV_SET_DELTA.

In the main ?PTRDEVICE packet, specify the window for which you are setting the pointer delta, and provide a word pointer to the ?PTRDEV_SET_DELTA subpacket.

In the ?PTRDEV_SET_DELTA subpacket, you specify the number of pixels the pointer must move in order to generate a movement event. You can specify different values for movement along the x- and y-axes. Figure 2-177 shows the subpacket structure; Table 2-158 details its contents.

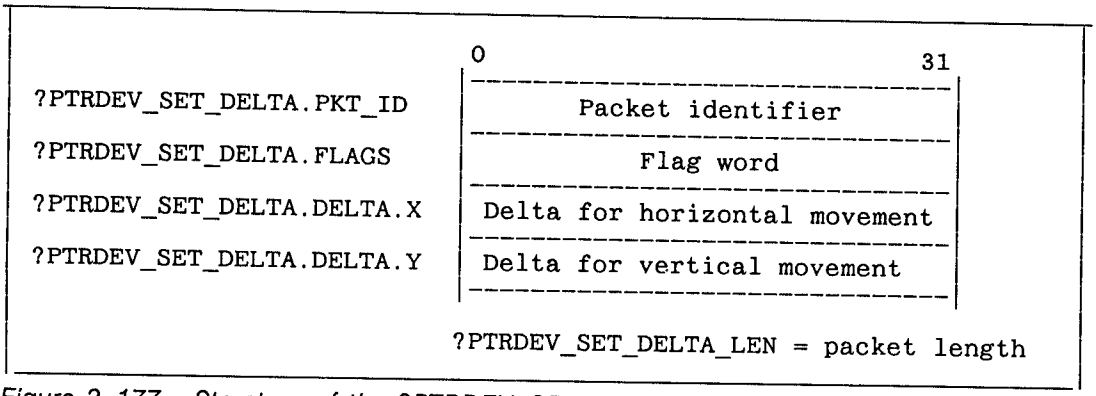


Figure 2-177. Structure of the ?PTRDEV_SET_DELTA Subpacket

Table 2-158. Contents of the ?PTRDEV_SET_DELTA Subpacket

Offset	Contents
?PTRDEV_SET_DELTA.PKT_ID (double word)	Packet identifier; set to ?PTRDEV_SET_DELTA_PKTID.
?PTRDEV_SET_DELTA.FLAGS (double word)	Flag word. If you are supplying a value, you must set the corresponding flag. ?PTRDEV_SET_DELTA.FLAGS.X -- Set this flag if you are specifying a delta for horizontal movement; set offset ?PTRDEV_SET_DELTA.DELTA.X to the delta value you want. ?PTRDEV_SET_DELTA.FLAGS.Y -- Set this flag if you are specifying a delta for vertical movement; set offset ?PTRDEV_SET_DELTA.DELTA.Y to the delta value you want.
?PTRDEV_SET_DELTA.DELTA.X (double word)	Number of pixels the pointer must travel horizontally in order to cause a movement event. This value must be an unsigned 32-bit value greater than 0. (You must also set the flag ?PTRDEV_SET_DELTA.FLAGS.X in the flag word ?PTRDEV_SET_DELTA.FLAGS.)
?PTRDEV_SET_DELTA.DELTA.Y (double word)	Number of pixels the pointer will travel vertically in order to cause a movement event. This value must be an unsigned 32-bit value greater than 0. (You must also set the flag ?PTRDEV_SET_DELTA.FLAGS.Y in the flag word ?PTRDEV_SET_DELTA.FLAGS.)

Getting Information About the Last Pointer Event

?PTRDEV_LAST_EVENT returns information about the last pointer event that was generated for a particular window.

In the main ?PTRDEVICE packet, specify the window for which you want the last event, and provide a word pointer to the ?PTRDEV_LAST_EVENT subpacket. The operating system returns information in the subpacket (see Figure 2-178).

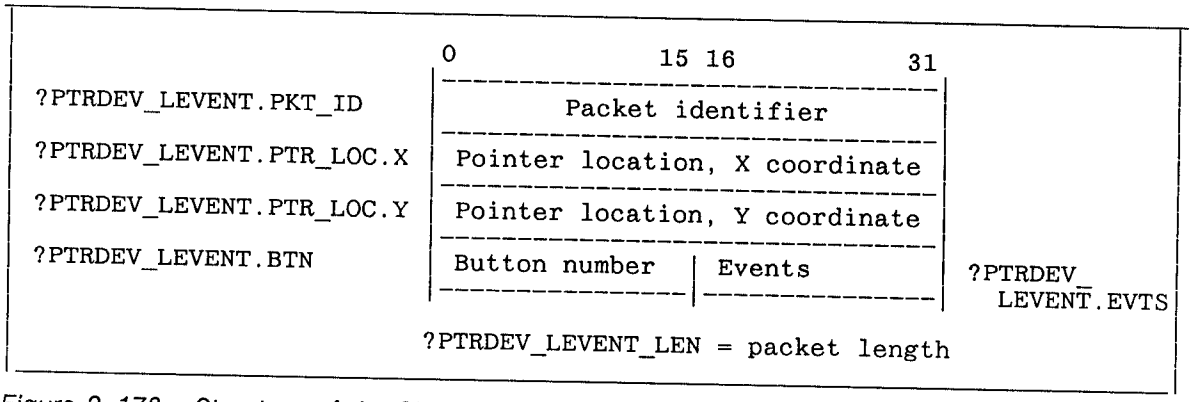


Figure 2-178. Structure of the ?PTRDEV_LAST_EVENT Subpacket

Table 2-159. Contents of the ?PTRDEV_LAST_EVENT Subpacket

Offset	Contents
?PTRDEV_LEVENT.PKT_ID (double word)	Packet identifier; set to ?PTRDEV_LEVENT_PKTID.
?PTRDEV_LEVENT.PTR_LOC.X (double word)	X coordinate of pointer location at the time the last event occurred; in pixels, relative to the origin of the window's virtual terminal (Returned by the operating system.)
?PTRDEV_LEVENT.PTR_LOC.Y (double word)	Y coordinate of pointer location at the time the last event occurred; in pixels, relative to the origin of the window's virtual terminal (Returned by the operating system.)
?PTRDEV_LEVENT.BTN	The number of the button involved in the event (if any). If the last event did not involve a button, this offset is set to -1. (Returned by the operating system.)
?PTRDEV_LEVENT.EVTS	The type of pointer event that occurred (returned by the operating system). If no event has yet occurred, this offset is set to -1. Event types are ?PTRDEV_EVENTS_MOVEMENT — Movement ?PTRDEV_EVENTS_BTN_DOWN — Button down ?PTRDEV_EVENTS_BTN_UP — Button up ?PTRDEV_EVENTS_ENTER_WINDOW — Enter window ?PTRDEV_EVENTS_EXIT_WINDOW — Exit window ?PTRDEV_EVENTS_ACTIVATION — Activation ?PTRDEV_EVENTS_DEACTIVATION — Deactivation ?PTRDEV_EVENTS_DBL_CLICK — Double click

Controlling the Appearance of the Pointer

You use ?PTRDEVICE, function code ?PTRDEV_SET_POINTER, to control how the pointer looks when it is within a particular window. You can

- Select a shape from the predefined pointer shapes.
- Make the pointer visible or invisible.

In the main ?PTRDEVICE packet, specify the window for which you want to set the pointer, and provide a word pointer to the ?PTRDEV_SET_POINTER subpacket. Any changes you make to the pointer will be in effect only when the pointer is within the window you specify.

Figure 2-179 shows the structure of the ?PTRDEV_SET_POINTER subpacket; Table 2-160 details its contents.

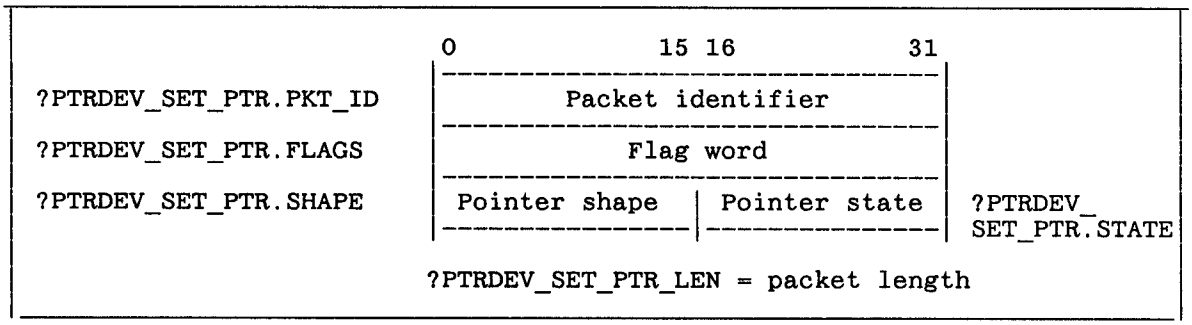


Figure 2-179. Subpacket for ?PTRDEV_SET_POINTER Subpacket

Table 2-160. Contents of the ?PTRDEV_SET_POINTER Subpacket

Offset	Contents
?PTRDEV_SET_PTR.PKT_ID (double word)	Packet identifier; set to ?PTRDEV_SET_PTR_PKTID.
?PTRDEV_SET_PTR.FLAGS (double word)	Flag word. Indicates which pointer attributes you are changing. ?PTRDEV_SET_PTR.FLAGS.SHAPE -- Set this flag if you are changing the pointer shape. ?PTRDEV_SET_PTR.FLAGS.STATE -- Set this flag if you are changing the state of the pointer (making it visible or invisible).
?PTRDEV_SET_PTR.SHAPE	The pointer shape you want. Available shapes are ?PTRDEV_PTR_SHAPE_DEFAULT The operating system determines shape. ?PTRDEV_PTR_SHAPE_ARROW An arrow. ?PTRDEV_PTR_SHAPE_FINGER A pointing finger. ?PTRDEV_PTR_SHAPE_SXHAIR A small cross-hair. ?PTRDEV_PTR_SHAPE_FXHAIR A full-screen cross-hair. ?PTRDEV_PTR_SHAPE_HOURGLASS An hourglass.
?PTRDEV_SET_PTR.STATE	The state of the pointer (visible or invisible). By default, the pointer is visible. ?PTRDEV_PTR_STATE_ENABLE Set this flag to make the pointer visible. ?PTRDEV_PTR_STATE_DISABLE Set this flag to make the pointer invisible.

Getting the Pointer Status

You can use ?PTRDEVICE, function code ?PTRDEV_GET_PTR_STATUS, to get the following information about the current state of the pointer:

- The current delta values.
- The events you have requested.
- The buttons you have requested.
- The current pointer shape.
- The current state of the pointer: enabled (visible) or disabled (invisible).

In the main ?PTRDEVICE packet, specify the window for which you want the pointer status, and provide a word pointer to the ?PTRDEV_GET_PTR_STATUS subpacket. The operating system returns information in the subpacket. Figure 2–180 shows the subpacket structures; Table 2–161 details its contents.

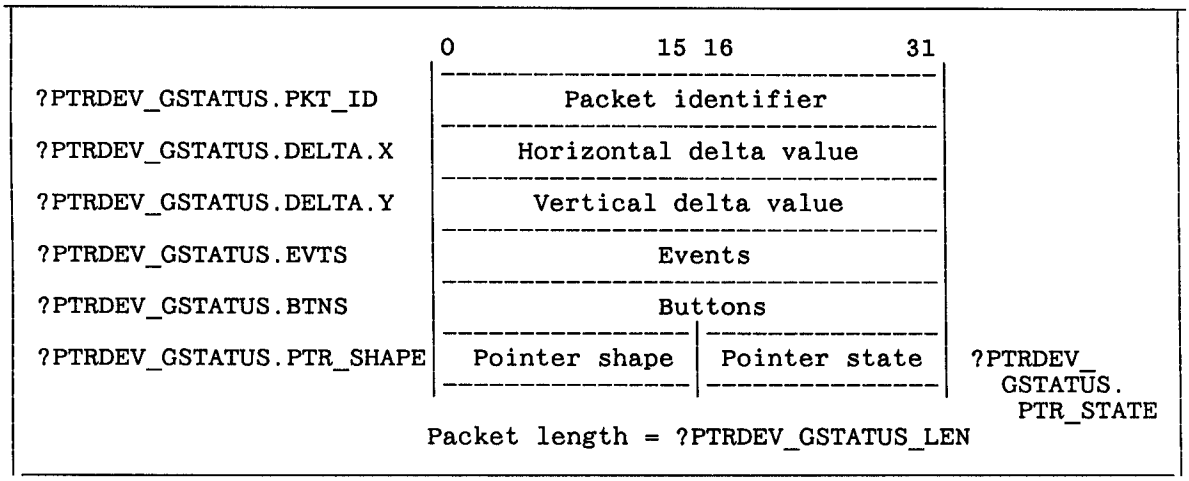


Figure 2–180. Structure of the ?PTRDEV_GET_PTR_STATUS Subpacket

Table 2-161. Contents of the ?PTRDEV_GET_PTR_STATUS Subpacket

Offset	Contents
?PTRDEV_GSTATUS.PKT_ID (double word)	Packet identifier. Set to ?PTRDEV_GSTATUS_PKTID.
?PTRDEV_GSTATUS.DELTA.X (double word)	Delta value for horizontal movement, in pixels. (Distance the pointer must move horizontally in order to cause a movement event.)
?PTRDEV_GSTATUS.DELTA.Y (double word)	Delta value for vertical movement, in pixels. (Distance the pointer must move vertically in order to cause a movement event.)
?PTRDEV_GSTATUS.SHAPE	The current pointer shape. Possible shapes are ?PTRDEV_PTR_SHAPE_ARROW — An arrow. ?PTRDEV_PTR_SHAPE_FINGER — A pointing finger ?PTRDEV_PTR_SHAPE_SXHAIR — A small cross-hair. ?PTRDEV_PTR_SHAPE_FXHAIR — A full-screen cross-hair. ?PTRDEV_PTR_SHAPE_HOURGLASS — An hourglass.
?PTRDEV_GSTATUS.STATE	Whether the pointer is visible (enabled) or invisible (disabled). By default, the pointer is visible (enabled). ?PTRDEV_PTR_STATE_ENABLE — The pointer is visible. ?PTRDEV_PTR_STATE_DISABLE — The pointer is invisible.

Licensed Material - Property of Data General Corporation

Moving the Pointer

You can use the ?PTRDEV_GENERATE_EVENT function to move the pointer within a window in the active group. The system moves the pointer to the location you specify, and places a pointer event in your input buffer that describes the pointer movement.

For example, suppose you use this function to move the pointer to location (50,50) in your application's window. In addition to moving the pointer, the system places a pointer event in the window's input buffer. This event is identical to a normal pointer event, except that it was generated via a system call instead of being caused by a user's action. The event tells your application that a movement event has taken place in your window, and that the pointer is now at location (50,50).

Because the pointer events you generate with this function are identical to those generated by a user's action with a physical pointer device, you can also use ?PTRDEV_GENERATE_EVENT to simulate pointer movement and button clicks. ?PTRDEV_GENERATE_EVENT is particularly useful for demonstrating and/or testing programs that read and interpret pointer event data.

NOTE: If the user has an absolute pointer device, such as a bit pad, ?PTRDEV_GENERATE_EVENT can change the pointer position only temporarily. Although the pointer does move to the position you specify, when the user next moves the pointer device, the pointer jumps back to the absolute position indicated by the pointer device.

In the main ?PTRDEVICE packet, specify the window for which you want to generate a pointer event, and provide a word pointer to the ?PTRDEV_GENERATE_EVENT subpacket.

In the ?PTRDEV_GENERATE_EVENT subpacket, you specify

- The type of pointer event you want to generate: movement, button up, button down, or double click.
- The coordinates to which you are moving the pointer (if you are generating a movement event).
- Whether the coordinates you are specifying are relative to the origin of the virtual terminal or to the origin of the physical screen.
- The button number (if you are generating a button–related event).

Figure 2–181 shows the subpacket structure; Table 2–162 details its contents.

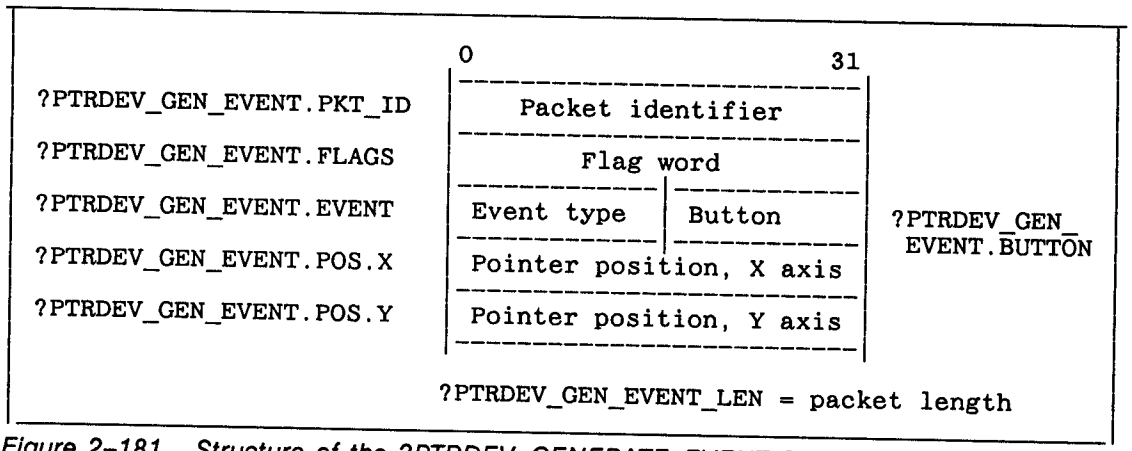


Figure 2–181. Structure of the ?PTRDEV_GENERATE_EVENT Subpacket

Licensed Material - Property of Data General Corporation

Table 2-162. Contents of the ?PTRDEV_GENERATE_EVENT Subpacket

Offset	Contents
?PTRDEV_GEN_EVENT.PKT_ID (double word)	Packet identifier: set to ?PTRDEV_GENERATE_EVENT_PKT_ID.
?PTRDEV_GEN_EVENT.FLAGS (double word)	Flag word. Flags are ?PTRDEV_GEN_EVENT.FLAGS.X -- Set this flag if you are specifying a new X coordinate for the pointer position. If you are not changing the X coordinate, set both this flag and the offset to 0. ?PTRDEV_GEN_EVENT.FLAGS.Y -- Set this flag if you are specifying a new Y coordinate for the pointer position. If you are not changing the Y coordinate, set both this flag and the offset to 0. ?PTRDEV_GEN_EVENT.VIRTUAL -- Set this flag if you are specifying a pointer position relative to the origin of the virtual terminal. (The position you specify on the virtual terminal must be visible.) ?PTRDEV_GEN_EVENT.PHYSICAL -- Set this flag if you are specifying a pointer position relative to the origin of the physical screen.
?PTRDEV_GEN_EVENT.EVENT	The type of event you want to generate. Event types are ?PTRDEV_EVENTS_MOVEMENT Movement ?PTRDEV_EVENTS_BTN_UP Button up ?PTRDEV_EVENTS_BTN_DOWN Button down ?PTRDEV_EVENTS_BTN_UP Double click
?PTRDEV_GEN_EVENT.BUTTON	The number of the button (if any) involved in the event. For a movement event, you must set this field to 0.
?PTRDEV_GEN_EVENT.POS.X (double word)	X coordinate of the desired pointer position. If relative to the origin of the physical screen, specify in pixels. If relative to the origin of the virtual terminal, specify in units appropriate to the virtual terminal type.
?PTRDEV_GEN_EVENT.POS.Y (double word)	Y coordinate of the desired pointer position. If relative to the origin of the physical screen, specify in pixels. If relative to the origin of the virtual terminal, specify in units appropriate to the virtual terminal type.

?PWDCRYP

Performs a password data encryption request.

?PWDCRYP *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERVBP	Invalid byte pointer passed as an argument
ERRVN	Reserved value not zero
ERPVS	Packet revision not supported
ERPRE	Invalid parameter passed as system call argument

Why Use It?

You can use ?PWDCRYP to add a level of security to the system password file. By using the ?PWDCRYP system call, the plain text password input to the call will be encrypted and returned to the caller. This system call could be used by products wishing secure system storage of various user profiles at a higher level functional layer than that of the operating system.

What It Does

?PWDCRYP performs a one way password encryption which uses the password as both the plain text and the "key" in the algorithm. The algorithm is based on the Data Encryption Standard (DES) algorithm proposed by the National Bureau of Standards (NBS).

?PWDCRYP performs the following function on behalf of the calling process:

- Perform a one-way password encryption of a plain text password whose byte address is contained in offset ?PWBP of the packet. The candidate password must be 1 to 32 bytes in length and be null terminated in the buffer. The input null terminator will not be encrypted.

Before issuing ?PWDCRYP, set the flags word (?PWFW) according to the function to be performed (see Table 2-163), and set the remaining offsets appropriately (see Figure 2-182). The operating system will return the length in bytes of the null terminated encrypted password. The length of the packet is ?PWSZ.

Figure 2-182 shows the structure of the ?PWDCRYP packet, and Table 2-163 describes the contents of each offset.

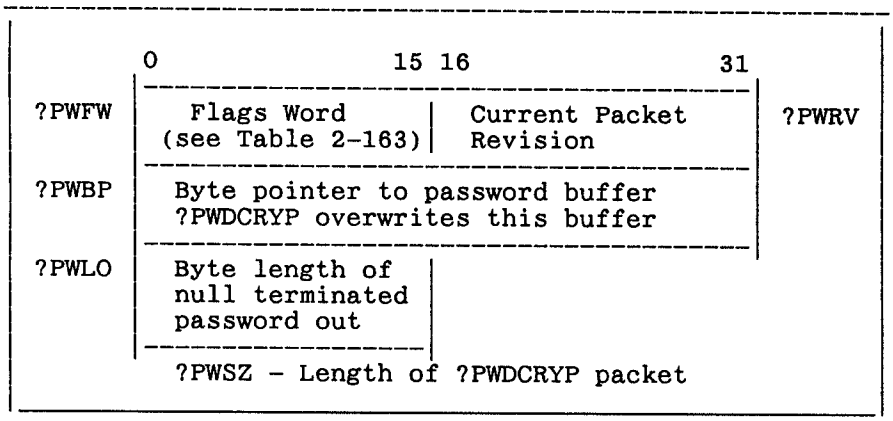


Figure 2-182. Structure of ?PWDCRYP Packet

?PWDCRYP Continued

Table 2-163. Contents of ?PWDCRYPT Packet

Offset	Contents
?PWFW	Flags word contains the following flags: ?PWOW--perform a one-way password encryption (Flag ?PWOW must be set on input to ?PWDCRYP.)
?PWRV	Current packet revision number; should be zero.
?PWBP	Byte address of password buffer containing the password in your logical address space. NOTE: The password buffer length must be at least 33 bytes in length.
?PWLO	Length in bytes of the output password that is null terminated. This is returned by the OS.

?PWRB

Performs physical block I/O.

?PWRB *[packet address]*
error return
normal return

See the description of ?PRDB in this chapter for further information on ?PWRB.

?RCALL Releases one resource and acquires a new one. (16–Bit Processes Only)

?RCALL *[procedure entry]*
normal return
alternate return

Input

Input accumulators and carry bit are passed to the new procedure

Output

Accumulators and carry bit are unchanged, unless the new procedure modifies them

Error Codes in AC0

The following error codes may be passed to the ?BOMB routine:

ERICM	Illegal system command
ERLRF	Overlay load error
ERSEN	Invalid shared library reference (The operating system does not support shared libraries.)

Why Use It?

?RCALL and the other resource system calls (?KCALL and ?RCHAIN) allow you to augment a 16–bit process's address space by acquiring disk–resident resources. You can use the resource system calls to acquire both root and overlay resources.

What It Does

?RCALL releases the calling resource, loads a new resource, and transfers control to the procedure entry that you specify within the new resource. You can pass the procedure entry as an argument to ?RCALL or pass it on the stack (with the .PTARG pseudo–op). If you pass the procedure entry on the stack, the operating system pops it off the stack as it executes ?RCALL.

After the operating system executes the new target procedure, it returns control to the ?RCALL normal return, unless the target specifies an alternate return by issuing an ISZ ?ORTN,3 instruction followed by a RTN instruction.

If the calling resource is an overlay in a multiple overlay area, it must be position–independent and movable. Otherwise, the return address may be invalid.

Notes

- See the descriptions of ?KCALL and ?RCHAIN in this chapter.

?RCHAIN

Chains to a new procedure. (16–Bit Processes Only.)

?RCHAIN *[procedure entry]*

Input

Input accumulators and carry bit are passed to the new procedure

Output

Accumulators and carry bit are unchanged, unless the new procedure modifies them

Error Codes

The following codes may be passed to the ?BOMB routine:

ERICM	Illegal system command
ERLRF	Overlay load error
ERSEN	Invalid shared library reference (AOS/VS and AOS/DVS do not support shared libraries.)

Why Use It?

?RCHAIN allows you to chain from a ?KCALLed or ?RCALLed procedure to another procedure. Only procedures that have been ?KCALLed or ?RCALLed can issue ?RCHAIN. Typically, you use ?RCHAIN to connect separate sequential pieces of a large resource.

What It Does

Like ?RCALL, ?RCHAIN releases the calling procedure, loads a new resource, and transfers control to the procedure entry that you specify within the new resource. You can pass the procedure entry as an argument to ?RCHAIN or pass it on the top of the stack, via the .PTARG pseudo-op.

Unlike ?RCALL, ?RCHAIN is issued from one called procedure to another and, therefore, has no normal return. Although each procedure must begin with an ?RSAVE instruction, only the last procedure on the chain should end with a RTN. When the operating system executes the RTN, it transfers control back to the initial procedure, not to the ?RCHAIN caller.

Notes

- See the descriptions of ?KCALL and ?RCHAIN in this chapter.

?RDB/?WRB Performs (reads/writes) block I/O.

?RDB *[packet address]*
error return
normal return

?WRB *[packet address]*
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Target file's channel number
AC2	Address of the ?RDB/?WRB 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, unless offset ?PRNH is -1 (See "Special Magnetic Tape Considerations" in this description.)
AC2	Address of the ?RDB/?WRB packet

Error Codes in AC0

ERDIO	Attempt to issue MCA direct I/O with outstanding requests
EREOF	End of file
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?RDB and ?WRB allow you to perform block I/O on a file. (A system call named ?BLKIO also lets you perform block I/O on a file, plus includes functionality for reading the next allocated element in a file.)

What It Does

?RDB and ?WRB respectively read and write blocks of data on magnetic tape, disk, or MCA links. Note that you cannot use ?RDB to read blocks into the write-protected area of your logical address space. However, you can use ?WRB to write from the write-protected area.

Before you issue ?RDB or ?WRB, load AC1 with the channel number that the operating system assigned to the target file when you opened it (?GOPEN), and define the ?RDB or ?WRB packet in your logical address space. You can load the packet address into AC2 before you issue the block I/O system call, or you can specify the packet address as an argument to ?RDB or ?WRB. The value for the length of the packet is ?PBLT. Figure 2-183 shows the structure of the block I/O packet, and Table 2-164 describes the contents of each offset.

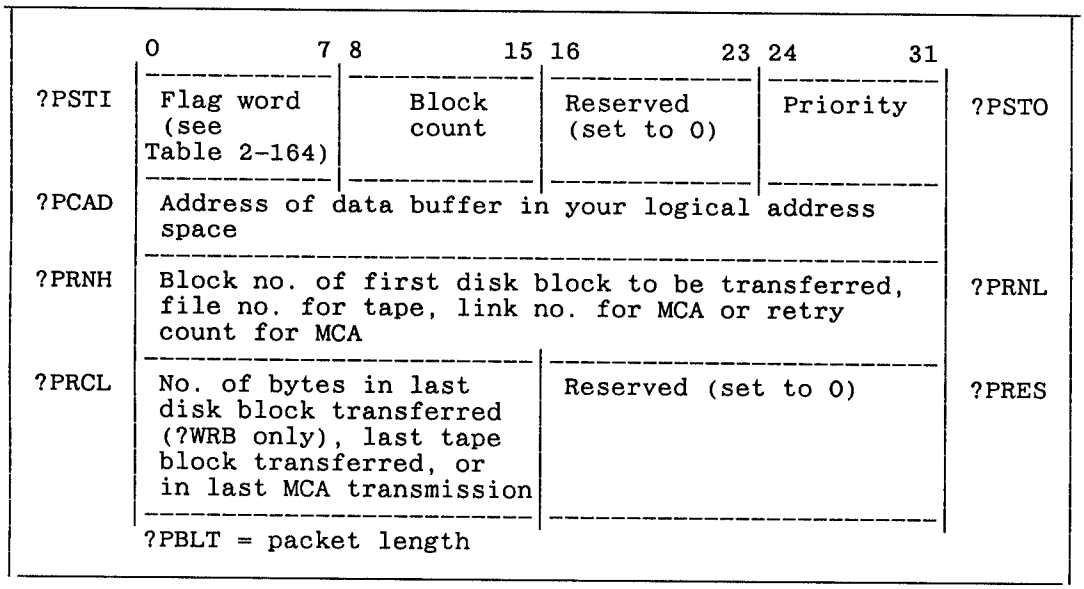


Figure 2-183. Structure of ?RDB/?WRB Packet

Before you can issue ?RDB or ?WRB against a file, you must open it with ?GOPEN.

Table 2-164. Contents of ?RDB/?WRB Packet*

Offset	Contents
?PSTI	Left byte contains the following flags: ?IMIO--Suspend MCA protocol. ?ENOV--Override logical end-of-tape or enable vertical format unit (VFU). ?SAFM--Safe ?WRB request for magnetic tape. Right byte contains the block count.
?PSTO	Left byte is reserved. (Set to 0.) Right byte contains the priority.
?PCAD (double word)	Address of data buffer in your logical address space.
?PRNH	Number of first disk block transferred (high-order bits) file number for tape, or line number for MCA.
?PRNL	Low-order bits of disk file block number (?PRNH), block number or -1 for tape, or retry count for MCA.
?PRCL	Number of bytes in the last disk block transferred (for ?WRB only), number of bytes in last tape block transferred, or number of bytes in last MCA transmission.
?PRES	Reserved. (Set to 0.)

* There is no default unless otherwise specified.

Disk Blocks

Specify the number of blocks you want to read or write in the right byte of offset ?PSTI. Bit ?IMIO in the left byte of ?PSTI applies only to MCA transfers. Bit ?ENOV in the same offset applies only to magnetic tape transfers. For more information about these options, see "Special Magnetic Tape Considerations" and "Special MCA Considerations."

Offset ?PCAD in the packet should point to the data buffer you reserved in your logical address space for the block I/O transfer. If you read or write disk blocks, use offsets ?PRNH and ?PRNL to indicate the relative block number of the first disk block you want to transfer.

If you write disk blocks (?WRB), use offset ?PRCL to indicate the number of bytes in the last block you want to transfer. As in the physical block I/O packet, this value indicates the last valid byte in the block. If you issue ?WRB to extend the file, the operating system places the end-of-file mark immediately after this byte. For disk block transfers, offset ?PRCL applies only to ?WRB. 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).

Special Magnetic Tape Considerations

If you perform block I/O on magnetic tapes, you have the option of specifying the file number at ?GOPEN time (for example, @MTB0:3) or in offset ?PRNH of the ?RDB/?WRB packet. If you used the CLI MOUNT command to mount the tape, you must ?GOPEN the file using its linkname. Offsets ?PRNH and ?PRNL specify a block number when you've specified the file number at ?GOPEN time.

When you ?GCLOSE the file, the operating system follows the last block written on a tape file with an end-of-file mark (two consecutive tape marks). By selecting the appropriate options in the block I/O packet, you can write beyond the end-of-tape mark and perform primitive tape functions, such as file positioning and rewinding.

To write beyond the end-of-tape mark, set offset ?PRNH (file number) to the highest file number plus one. Set the Block Count parameter (offset ?PSTI, right byte) to 0, and select bit ?ENOV (Bit 1) in offset ?PSTI. (Note that you must set ?ENOV to write beyond the end-of-tape mark.) Then, the operating system automatically follows the appended data with a new end-of-tape mark. To rewind a tape, set offsets ?PRNH, ?PRNL, and ?PSTI to 0.

When the operating system encounters an end-of-tape (EOT) mark while it is performing an ?RDB operation, it transfers the data it was able to read to the buffer designated by ?PCAD, and returns the byte count of that data to AC1. The byte count in AC1 is always a true value; that is, it is not rounded. The operating system then takes the error return from ?RDB, and returns error code EREOF to AC0.

Offset ?PRCL of the ?RDB/?WRB packet is the Block Length parameter for block I/O operations on magnetic tapes. If the operating system finds a block larger than this value during an ?RDB, it truncates the block to the length that you specify in ?PRCL.

The operating system always rounds the value of ?PRCL to an even number of bytes to force transfers to the buffer to be word aligned. The operating system discards the byte that exceeds the requested block size. If a block consists of an odd number of bytes on input, the operating system returns an even number of bytes on output. In this case, the last byte is undefined.

To position a tape at the end of a particular file, specify the file number in offset ?PRNH of the block I/O packet, and set offset ?PRNL (block number) to -1. To append data to the file, set the block count parameter (?PSTI, right byte) to a nonzero value. (You can append only on a ?WRB.) If you set the block count parameter to 0, the operating system returns the total number of blocks transferred to AC1.

Select the ?SAFM option in offset ?PSTI to write a logical end-of-tape (LEOT) or to write data followed by LEOT. By selecting this option, you ensure that the tape is always readable, even if you do not issue ?GCLOSE. This means that if your system fails, your tape will still be readable.

Special MCA Considerations

MCA processes in your system must be resident. You begin block I/O operations on MCAs by issuing ?GOPEN to open the MCA as a unit or to open a specific link. To open the MCA as a unit, load AC0 with a byte pointer to one of the following filenames before you issue ?GOPEN:

@MCAT
@MCAR
@MCAT1
@MCAR1

where

- @MCAT is an MCA to use for writing (transmitting).
- @MCAR is an MCA to use for reading (receiving).
- 1 is the second MCAT or MCAR device.

If you open the MCA as a unit, you must name a specific MCA link in offset ?PRNH of the block I/O packet for the ?RDB or ?WRB operation.

To open a specific MCA link, load AC0 with one of the following pathnames before you issue ?GOPEN:

@MCAT:n
@MCAR:n
@MCAT1:n
@MCAR1:n

where

- n is the link number (range 0 through 15).

Even if you specify the link number when you issue ?GOPEN, you can change it for each ?RDB or ?WRB system call by altering the value of ?PRNH in the ?RDB/?WRB packet. A link number of 0 applies to ?RDB system calls only, and directs the operating system to accept transmissions from any MCAT. The operating system satisfies read requests from link 0 after it performs all other read requests from specific link numbers.

When you close an MCA file previously opened for a specific link (that is, you specified the link number with ?GOPEN), the operating system transmits an end-of-file (EREOF) condition with a word count of 0 to the MCA receiver (after a 2-millisecond time-out). Do not use the link format unless the remote MCA is prepared for this transmission protocol.

The retry count, offset ?PRNL in the block I/O packet, applies only to ?WRB operations from an MCAT. You can set this parameter to any value within the range 0 through 255. Each count represents 20 attempts to establish communications with the target MCAR. A retry value of 0 directs the operating system to perform the maximum number of retries.

You can circumvent the MCA protocol and perform direct MCA I/O by selecting bit ?IMIO in offset ?PSTI. This method results in fewer interrupts. To use the direct mode, however, you must perform your own data and line validations. In addition, the operating system does not honor direct MCA transmission requests unless the MCAT's output queue is empty. If you try to transmit when the queue is not empty, the block I/O system call fails and the operating system returns error code ERPRO to AC0.

Similarly, once the operating system has queued a direct write request to an MCAT link, it does not queue any further direct I/O or protocol I/O requests until it completes the current request. If you try to perform a read or write on an MCA link while direct I/O is in progress, the system call fails, and the operating system returns error code ERDIO in AC0.

*

Data Channel Line Printers

If you perform block writes to a data channel line-printer controller with a DVFU unit, set bit ?ENOV in offset ?PSTI of the ?WRB packet. The DVFU (direct-access vertical forms control unit) is a hardware device that interfaces with the operating system's Forms Control Utility (FCU) program. By issuing VFU commands through the FCU, you can control the printer's horizontal and vertical formatting. A memory capability within the DVFU eliminates the need for mechanical formatting with a VFU tape.

If you omit ?ENOV, the operating system ignores all VFU definition commands.

For a complete description of the FCU utility, refer to your operating system's Command Line Interpreter (CLI) user's manual.

Sample Packet

```
PKT:    .BLK    ?PBLT           ;Allocate enough space for the packet.
                                ;Packet length = ?PBLT.
        .LOC    PKT+?PSTI       ;Number of blocks to read or write.
        .WORD   4               ;Transfer four blocks.
        .LOC    PKT+?PSTO       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?PCAD       ;Word address of data buffer.
        .DWORD  BUFF            ;Data buffer address is BUFF.
        .LOC    PKT+?PRNH       ;Block number.
        .DWORD  0
        .LOC    PKT+?PRCL       ;Actual number of bytes transferred.
        .WORD   0               ;The O/S returns this value.
        .LOC    PKT+?PRES       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?PBLT       ;End of packet.
BUFF:   .BLK    4*256           ;Text buffer area.
```

Notes

- See the description of ?GOPEN and ?BLKIO in this chapter.

?RDUDA/?WRUDA — Reads/writes a user data area (UDA).

?RDUDA *[packet address]*
error return
normal return

?WRUDA *[packet address]*
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the pathname of the file associated with the UDA
- 0 if a packet address is supplied

AC1 Reserved (Set to 0.)

AC2 One of the following:

- Address of a 128-word UDA receive buffer if not using a packet
- Address of the ?RDUDA/?WRUDA packet, unless you specify the address as an argument to ?RDUDA/?WRUDA

Output

AC0 Unchanged

AC1 Undefined

AC2 Unchanged or address of ?RDUDA/?WRUDA packet

Error Codes in AC0

ERFAD	File access denied
ERIFT	Illegal file type
ERNUD	User data area does not exist
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?WRUDA allows you to write to a file's UDA. The UDA frequently stores forms control parameters for files you intend to print on a data channel line printer that is controlled by the EXEC utility. (You can use a UDA for other purposes as well.) Because ?RDUDA returns the contents of a file's UDA, you can issue it to check the UDA contents before you issue ?WRUDA.

What It Does

?WRUDA writes to the target file's user data area (UDA). ?RDUDA reads the contents of the target file's UDA. The target file or directory can be specified in one of two ways: either by a byte pointer to the entry's pathname in AC0, or by using offset ?PUDCN in the ?RDUDA/?WRUDA packet. The use of the packet is only necessary if you choose to specify the target entry's channel number. AC0 must be set to 0 to indicate the use of a packet.

If not using a packet, load AC2 with a word pointer to a 128-word receive/send buffer from/for the UDA. If using a packet, offsets ?PUDAH and ?PUDAL must instead be used. The UDA must already exist. (You can use the ?CRUDA system call or the CLI Forms Control Utility to create a UDA.)

To find out whether a file has a UDA, issue ?FSTAT, and then examine bit ?FUDA in offset ?SSTS of the returned packet. If ?FUDA is set, the UDA exists.

When you write to a UDA, do not use codes that will set Bit 0 of the first UDA word (word 0). Data General reserves codes with this configuration for its own use.

Figure 2-184 shows the structure of the ?RDUDA/?WRUDA packet.

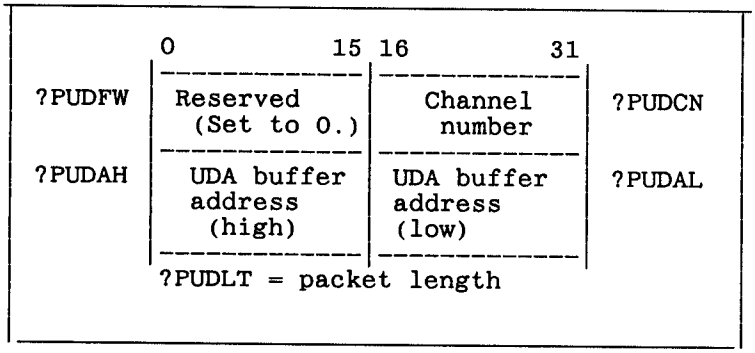


Figure 2-184. Structure of ?RDUDA/?WRUDA Packet

Notes

- See the descriptions of ?CRUDA and ?FSTAT in this chapter.

?READ/?WRITE Performs (reads/writes) record I/O.

?READ *[packet address]*
error return
normal return

?WRITE *[packet address]*
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Address of the I/O packet, unless you specify the address as an argument to the system call

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the I/O packet

Error Codes in AC0

EREOF	End of file
ERFIL	File read error
ERITP	Translation error (This applies to the selected field translation packet extension only.)
ERLTL	Line too long
ERRAD	Read access denied (both ?READ and ?WRITE)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument
ERWAD	Write access denied (?WRITE only)

What It Does

?READ and ?WRITE perform record I/O on an open file. To use ?READ, the calling process must have Read access to the file. To use ?WRITE, the calling process must have both Read and Write access to the file.

As Figure 2–185 shows, the basic ?READ/?WRITE packet has the same structure as the ?OPEN and ?CLOSE packets. Some specifications apply exclusively to ?READ and ?WRITE, while others apply to all record I/O system calls. When you issue ?READ or ?WRITE against a file, you can default some specifications to the values that you selected when you created or opened the file.

NOTE: If you are using a pixel-mapped system, do not attempt to use device code 10 (TTI or system console input, such as a keyboard) or device code 11 (TTO or system console output, such as a screen or printer), once your operating system is running.

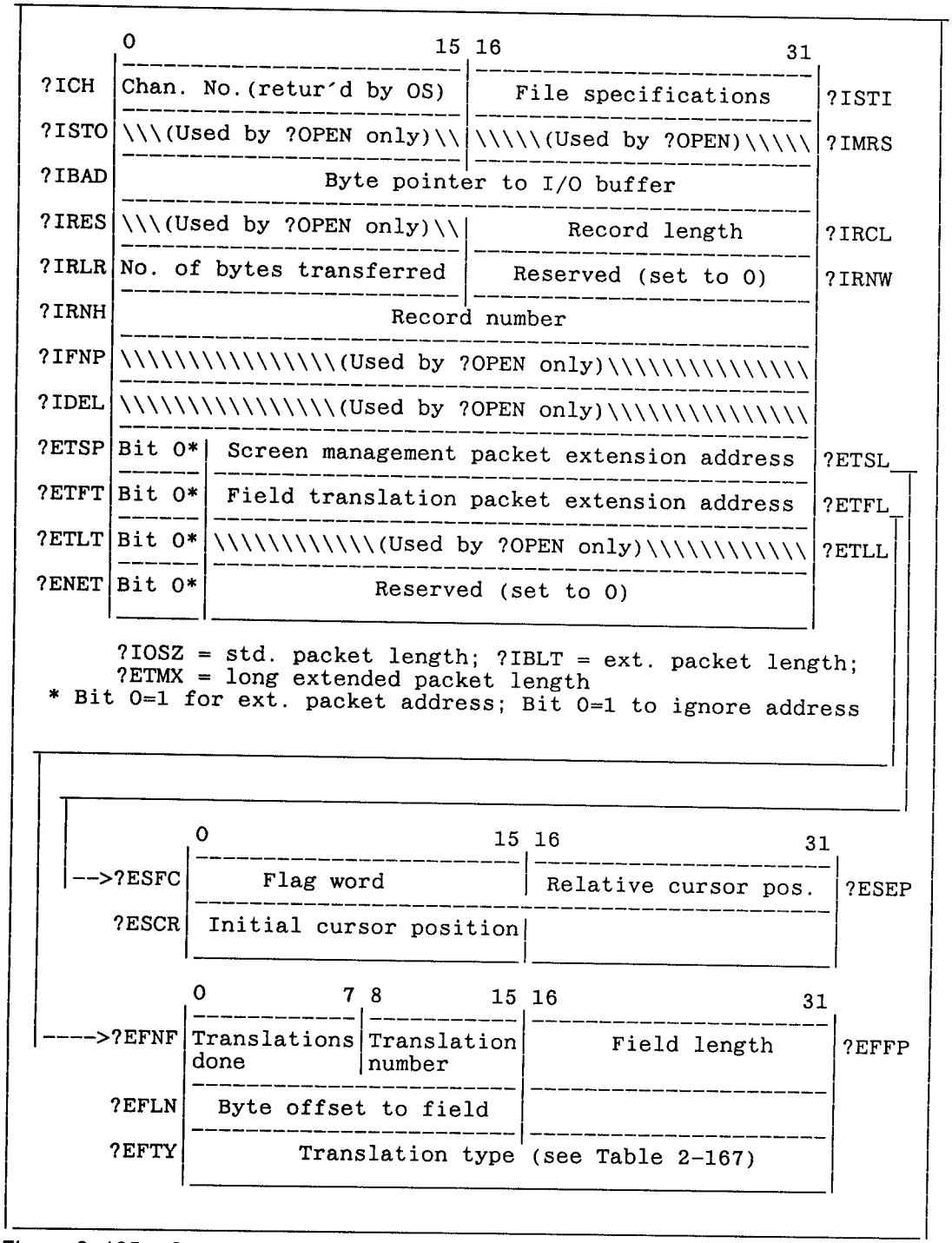


Figure 2-185. Structure of ?READ/?WRITE Packet

Similarly, you can use the ?READ and ?WRITE packets to set parameters you deferred when you created or opened the file. Figure 2-185 shows the structure of the ?READ/?WRITE packet. Table 2-165 describes each offset and parameter.

?ISTI: The File Specifications Word

Offset ?ISTI, the file specifications word, provides a number of options for the file you are reading or writing. To default an ?ISTI field, you must explicitly set that field to 0.

The ?ISTI specifications are bit masks. Therefore, if you want to select more than one ?ISTI specification, you must OR the bit masks. For example, the specification ?ICRF!?RTDS sets the record format (?ICRF) to data-sensitive (?RTDS).

To change the file's record format, set offset ?ICRF and one of the following masks in the record format field:

- ?RTDY (dynamic-length).
- ?RTDS (data-sensitive).
- ?RTFX (fixed-length).
- ?RTVR (variable-length).
- ?RTUN (undefined-length)..

  Use ?RTUN only if you are reading an unlabeled magnetic tape with an unknown block size, or if you are reading an IBM labeled tape with the undefined-length record format.

- ?RTVB (variable-block, variable-record)

  Use ?RTVB only if you are reading an IBM labeled tape with the variable block (spanned) record type.

The record format stays the same until you change it or close the file. You can also default this parameter, but only if you specified a record format when you created or opened the file. If a file has no record format parameter, the I/O system call fails, and the operating system returns an error to AC0.

Table 2-165. Contents of ?READ/?WRITE Packet*

Offset	Contents
?ICH	Channel number (returned by the OS).
?ISTI	File specifications. Packet type. ?IPKL--Indicates presence of an extended packet. DEFAULT = 0 (standard ?READ/?WRITE packet). Format select. ?ICRF--Change format to that specified in the record format field of this packet. DEFAULT = 0 (record format specified when the file was opened). File pointer positioning. ?IPST--Offset ?IRNH contains an absolute record number for fixed records and an absolute byte offset for other record types. DEFAULT = 0 (offset ?IRNH contains a relative record offset for fixed records and a relative byte offset for other record types). Append (used only by ?OPEN). Binary I/O. ?IBIN--Character device I/O is in binary mode. Binary mode is selected on each ?READ or ?WRITE request. This mode prevents the operating system from interpreting special characters and echoing the input data. DEFAULT = 0 (character device I/O is in text mode).

* There is no default unless otherwise specified.

(Continues)

Table 2-165. Contents of ?READ/?WRITE Packet*

Offset	Contents
?ISTI (Continued)	Force output. ?IFOP--Write buffer from system area. DEFAULT = 0 (normal system buffering). Exclusive open (used only by ?OPEN). Priority reads (used only by ?OPEN). Creation option (used only by ?OPEN). Input/output (used only by ?OPEN). Record format. ?RTDY--Dynamic-length. ?RTDS--Data-sensitive. ?RTFX--Fixed-length. ?RTVR--Variable-length. ?RTUN--Undefined-length. ?RTVB--Variable-block, variable-record (IBM labeled tapes only).
?ISTO	File type (used only by ?OPEN).
?IMRS	Physical block size (used only by ?OPEN).
?IBAD (double word)	Byte pointer to I/O buffer. DEFAULT = -1 (use the I/O buffer specified when the file was opened).
?IRES	Density mode (used only by ?OPEN).
?IRCL	Record length. States the number of bytes to read or write for dynamic-length and fixed-length records and the maximum number of bytes to read or write for data-sensitive and variable-length records. DEFAULT = -1 (use the record length specified when the file was opened).

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

Table 2-165. Contents of ?READ/?WRITE Packet*

Offset	Contents
?IRLR	Actual number of bytes transferred by this I/O call (returned by the operating system).
?IRNW	Reserved. (Set to 0.)
?IRNH (double word)	Record number. A double-precision integer that indicates the current record number. To position file pointer at beginning of file, set ?IRNH and ?IPST to 0. To position file pointer at end of file, set ?IRNH to -1 and IPST to 0. To position file pointer at the next sequential record, set ?IPST and set ?IRNH to 0.
?IFNP (double word)	Byte pointer to pathname (used only by ?OPEN).
?IDEL (double word)	Address of delimiter table (used only by ?OPEN).

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

Positioning the File Pointer

The file-pointer position parameter, ?IPST, determines how the operating system interprets the ?IRNH offset. ?IRNH contains a double-precision record number. These two parameters let you move the file pointer forward or backward. Normally, the file pointer points to the next record before each read or write. The file pointer parameters work together as follows:

- If you set ?IPST and the record format for the file is fixed-length (?RTFX), the operating system interprets the value in ?IRNH as an absolute record number (from the beginning of the file).
- If you set ?IPST and the record format is dynamic-length (?RTDY), data-sensitive (?RTDS), variable-length (?RTVR), variable-block, variable-record (?RTVB), or undefined-length (?RTUN), the operating system interprets the value in ?IRNH as an absolute byte count (from the beginning of the file).
- If you set ?IPST to 0 and the record format is fixed-length, the operating system interprets the value in ?IRNH as a record offset, relative to the file pointer's current position; the operating system begins the I/O transfer at this position in the file.
- If you set ?IPST to 0 and the record format is other than fixed-length, the operating system interprets ?IRNH as a byte offset, relative to the file pointer's current position; the operating system begins the I/O transfer at this position in the file.

You can also use ?IPST and ?IRNH to move the file pointer to either the beginning or the end of the file. To move the pointer to the beginning of the file, set ?IPST and specify 0 in ?IRNH. To move the pointer to the end of the file, set ?IPST and specify -1 in ?IRNH.

To simply read or write the next sequential record in the file (relative to the file pointer's current postion), set both ?IPST and the ?IRNH pair to 0.

The binary I/O parameter, ?IBIN, applies only to character devices, and directs the operating system to pass each byte from the device as is; that is, without interpreting it. When you read from a character device in binary mode, the operating system ignores rubout characters and all control sequences. The alternative to binary mode is text mode, in which the operating system interprets each byte based on the character device's characteristics. (Text mode is the default.) To switch from binary mode input to text mode input, do one of the following:

- Press the BREAK key on your console (that is, communicate over an ALM-8 or an ALM-16 communications line).
- Issue a text mode ?READ.
- Terminate the process that issued the ?READ.

Note that none of these actions affects binary mode output on a ?WRITE.

For console devices, the force output option, ?IFOP, negates the CTRL-O character you typed previously and writes the records to the console. For disk files, magnetic tape files, and MCA and IPC files, this option forces the operating system to write the current buffer from a dynamic system area to the file.

If you issue ?READ/?WRITE against a pipe file, the operating system ignores the following bits of offset ?ISTI in the packet:

- ?IPST and offset ?IRNH — relative or absolute file positioning.
- ?IBIN — setting binary I/O on the pipe.
- ?IIPC — setting "IPC No Wait" on the pipe.

Screen Management Primitives

In addition to the standard packet, the operating system recognizes two packet extensions for ?READ and ?WRITE: a screen management extension and a selected field translation extension.

You can access certain screen management primitives by appending the screen management extension to the standard ?READ or ?WRITE packet. These screen management primitives are normally controlled by the operating system's peripheral manager. For either ?READ or ?WRITE, this feature allows you to specify that the cursor position be returned. For ?READ, this feature lets you display an initial string, ignore typed-ahead characters, specify no echo, and enable screen-edit control characters.

Screen-edit control characters provide you with such text editing capabilities as character insertion and character overwrite, as well as the ability to position your cursor anywhere in the current buffer. Note, however, that the ability to delete characters is not a feature unique to screen edit. Also, you cannot use screen-edit control characters on a hard-copy console.

With both ?READ and ?WRITE, you can use the screen management extension to suppress echoing of delimiters and to specify an initial cursor position. The initial cursor position, which is expressed in column and row-number format, places the cursor at a specific position in the display before the I/O sequence occurs.

To use the screen-management extension, structure the ?READ and ?WRITE packet as follows:

- Set bit ?IPKL in offset ?ISTI of the I/O packet.
- Set Bits 1 through 31 of offset ?ETSP to the address of the screen-management extension.
- Set Bit 0 of ?ETSP to 1 (to indicate the presence of the extension).

Set up the screen-management extension elsewhere in your code.

The first time you use the screen-management extended packet, set Bit 0 of offset ?ETSP to 1, so that the operating system will recognize Bits 1 through 31 in offset ?ETSP as the address of the packet extension.

Licensed Material - Property of Data General Corporation

After the operating system executes ?READ or ?WRITE, it clears Bit 0 of ?ETSP. If you want to change the packet extension and reuse it, reset Bit 0 of ?ETSP to 1. Otherwise, the operating system assumes that the packet has not changed and, therefore, does not examine it again.

Figure 2-186 shows the structure of the screen-management packet extension. Table 2-166 describes each offset.

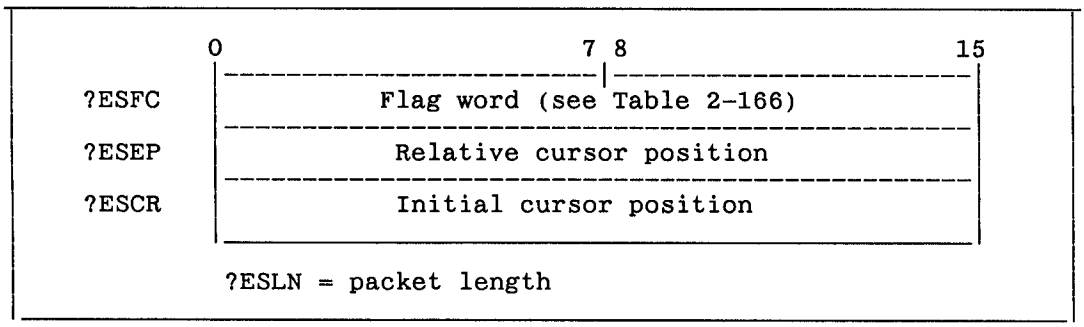


Figure 2-186. Structure of Screen-Management Packet Extension

You can also use screen-management primitives to enable autoterminating ?READ system calls. An autoterminating ?READ does not require you to enter a delimiter before the system call completes. Instead, it terminates itself. You can use an autoterminating ?READ to protect data that a previous ?READ or ?WRITE displayed on your screen.

Table 2-166. Contents of Screen-Management Packet Extension*

Offset	Contents
?ESFC	Flag word, with one or more of the following bit masks: ?ESSE--Screen edit; enable screen-edit control characters (CRT devices only). ?ESRD--Redisplay data that was read previously (?READ only). (Valid only if ?ESSE is also set.) ?ESNR--Drop typed-ahead characters; that is, flush the ring buffer before beginning input (?READ only). ?ESED--Do not echo delimiters. ?ESCP--Initial cursor position is specified in offset ?ESCR of the packet extension. ?ESDD--?READ ended with a double delimiter; this bit is set by the system. ?ESRP--Return cursor position after I/O. ?ESNE--Do not echo input. ?ESGT--Read typed-ahead characters, end ?READ if ring buffer is empty. ?ESBE--?READ ended because ring buffer is empty; this bit is set by the system. ?ESPE--The read terminated with a pointer event (valid only if ?ESSE is also set).
?ESEP	Relative position within the data to place the cursor before input, where the first character position is 0. (?READ only, valid only if both ?ESSE and ?ESRD are set.)
?ESCR	Initial cursor position, where the left byte of ?ESCR indicates the relative column and the right byte of ?ESCR indicates the relative row; the leftmost column, top row of the console is Column 0, Row 0. (This feature will not work correctly if you have not set the correct device characteristics for the console.

* There is no default unless otherwise specified.

NOTE: In order for ?READ to delimit on pointer events and thus receive flag ?ESPE, you should previously have set the ?TRPE characteristic via system call ?GECHR.

To terminate ?READ without an autoterminator, you must perform the following steps:

1. Make sure the data buffer that offset ?IRCL in the ?READ packet points to contains two consecutive ASCII Delete characters (<177><177>) at the end of the buffer.

   The number of character positions in the buffer before the first Delete character defines the length of the autoentry field.

2. Enter a full field of characters or type a delimiter.

   If you type a delimiter, the operating system stores that delimiter in the data buffer that is visible on your screen. (Set bit ?ESED in offset ?ESFC of the screen-management-primitives packet extension if you do not want the delimiter to echo on your screen.) For example, if you specify the predisplay option (bit ?ESRD), the delimiter will overwrite the first Delete character in the buffer. The same thing will happen if you issue a screen-edit control character to position the cursor at the end of the buffer.

   If you type a function key delimiter, the delimiter overwrites both Delete characters in the data buffer that is visible on on your screen.

To terminate ?READ with an autoterminator, you must perform the following steps:

1. Set bit ?ESSE in offset ?ESFC of the screen-management-primitives packet extension.

2. Make sure the data buffer that offset ?IRCL in the ?READ packet points to contains two consecutive ASCII Delete characters (<177><177>) at the end of the buffer.

   The number of character positions in the buffer before the first Delete character defines the length of the autoentry field.

3. Position the cursor over a Delete character at the end of the buffer.

4. Type any character that is not a screen-edit control character.

   The operating system stores that character in the data buffer that is visible on your screen. (Set bit ?ESED in offset ?ESFC of the screen-management-primitives packet extension if you do not want the character to echo on your screen.) Unlike a delimiter, however, an autoterminator will not overwrite the first Delete character in the buffer.

?READ/?WRITE Continued

You can also use screen management primitives to perform self-terminating ?READs. A self-terminating ?READ terminates itself after reading all the characters in the input buffer. It will not terminate, however, until at least one character has been read from the buffer. This type of ?READ is most useful if your application program can handle data as soon as it is entered, without waiting for a delimiter or a specific number of characters to be typed. It can be used as an alternative to single-byte I/O, or in an environment where data of varying amounts tends to arrive in spurts.

In order to perform a self-terminating ?READ you must do the following:

1. Supply a screen-management extension to the ?READ system call.
2. Set bit ?ESGT in the screen management packet.
3. If there is no data in the input ring buffer, the ?READ will wait for a character to be entered.
4. If there is data in the input ring buffer, the ?READ will take characters from the input ring buffer until it is empty, or until the condition to terminate the ?READ (entering a delimiter, for a data-sensitive ?READ or entering ?IRCL characters, for a fixed-length ?READ) is met.
5. If the ?READ terminates because there are no more characters in the input ring buffer, the system sets bit ?ESBE in the screen management packet extension.

Restriction: If you set bit ?ESGT in the screen management packet, you must NOT set bit ?ESSE.

Selected Field Translation

The selected field translation extension lets you translate one or more fields within a record during the I/O sequence. The translations this packet allows are ASCII lowercase to ASCII uppercase or vice versa, and ASCII to EBCDIC or vice versa. In addition, you can select one of the following three parity options for the record:

- Even parity
- Odd parity
- No parity (strip parity bits)

The procedure for using the selected field translation extension is similar to the procedure for the screen management extension. Briefly, you must perform the following steps:

1. Set bit ?IPKL in offset ?ISTI of the standard ?READ or ?WRITE packet.
2. Set Bits 1 through 31 of offsets ?ETFT to the address of the field translation packet.
3. Set Bit 0 of ?ETFT to 1 (to indicate the presence of the extension).

Supply the packet elsewhere in your program.

To reserve the correct number of words for this extension, you must begin it or end it with the pseudo-op

```
.LOC_extended_pkt_address+?EFNF+n
```

where

- n is the number of the subpacket * subpacket length (See the following for information on subpackets.)

Bit 0 of ?ETFT serves the same purposes as the flag bits for the other I/O packet extensions; that is, the operating system examines the packet extension only if the flag bit is set to 1. Because the operating system sets this bit to 0 after it executes the packet extension, you must reset this bit to 1 if you want to reuse the extension.

Figure 2–187 shows the structure of the packet extension for selected field translation, and Table 2–167 describes the contents of each offset.

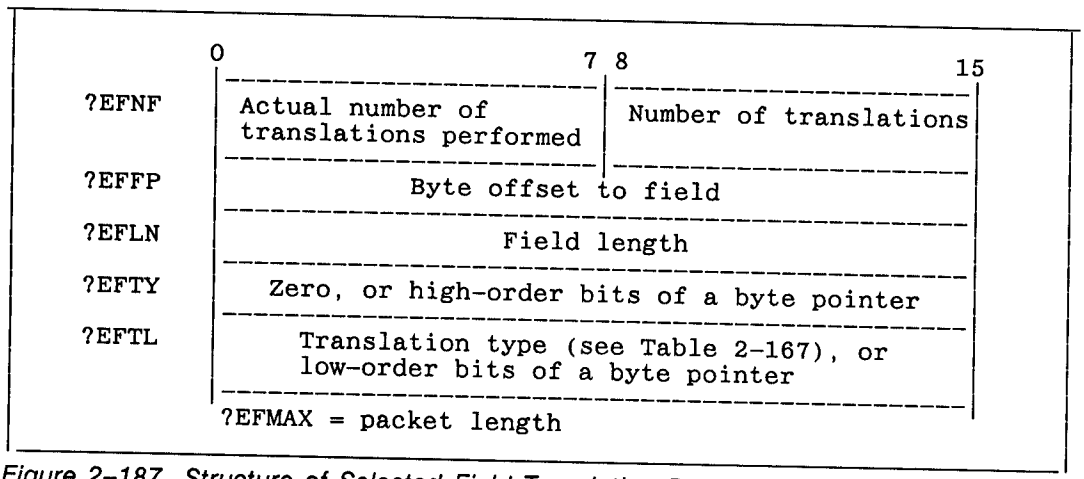


Figure 2–187. Structure of Selected Field Translation Packet Extension

Offsets ?EFFP, ?EFLN, ?EFTY, and ?EFTL form a subpacket within the field translation packet. You must use one subpacket for each translation you want to perform. Use the following pseudo-op after each subpacket to space past that subpacket:

```
.LOC subpacket_address+?EFMAX
```

Table 2-167. Contents of Selected Field Translation Packet Extension*

Offset	Contents
?EFNF	Right byte: Number of translations to perform. Left byte: Actual number of translations performed. (The OS returns this value.)
	?EFFP Byte offset to field (within the specified record).
	?EFLN Field length (in bytes). Default = 0 (translates actual number of bytes transferred)
	?EFTY/ ?EFTL Translation type.
Translation subpacket: one per translation	?ASEB--ASCII to EBCDIC. ?EBAS--EBCDIC to ASCII. ?ALAU--ASCII lowercase to ASCII uppercase. ?AUAL--ASCII uppercase to ASCII lowercase. ?SPAR--Strip parity bits. ?AEPR--Add even parity. ?AOPR--Add odd parity.
	NOTE: Instead of these masks, you can use offsets ?EFTY and ?EFTL as a byte pointer to a 256-byte translation table.

* There is no default unless otherwise specified.

The left byte of offset ?EFNF specifies the number of translations for this ?READ or ?WRITE and, therefore, the number of subpackets.

The operating system performs the translations in the order in which the subpackets appear. Thus, if you want to translate a field from EBCDIC to ASCII, and then add odd parity, specify the EBCDIC to ASCII translation in the first subpacket, and specify odd parity in the second subpacket.

Note that the translation tables work in only one direction. For example, if you want to translate an EBCDIC field to ASCII on a ?READ, and then write the records back in EBCDIC format, you must redefine the translation type before performing the ?WRITE.

The program excerpt in Figure 2-188 shows two translations on the same record.

```
PKT:    .LOC    PKT+?EFNF+8.      ;Right byte: No. of translations?
        .WORD   2                 ;Two.

        .LOC    PKT+?EFNF+6       ;Reserve space for two subpackets.

;"TRNS0" is the first subpacket (for first field translation).

TRNS0:  .LOC     TRNS0+?EFFP      ;Byte offset to field.
        .WORD   0                 ;Field starts with first byte.

        .LOC    TRNS0+?EFLN       ;Field length.
        .WORD   8.                ;Eight bytes (four 16-bit words).

        .LOC    TRNS0+?EFTY       ;Translation type
        .DWORD  ?ALAU             ;ASCII lowercase to uppercase.

        .LOC    TRNS0+?EFMAX      ;Space past first subpacket.

"TRNS1" is the second subpacket (for second field translation).

TRNS1:   .LOC    TRNS1+EFFP       ;Byte offset to field.
        .WORD   8.                ;Field starts with ninth byte.

        .LOC    TRNS1+?EFLN       ;Field length.
        .WORD   0                 ;Default (rest of record)

        .LOC    TRNS1+?EFTY       ;Translation type
        .DWORD  ?ALAU             ;ASCII lowercase to uppercase.

        .LOC    TRNS1+?EFMAX      ;Space past second subpacket.
```

Figure 2-188. Selected Field Translation Packet Sample Listing

Licensed Material - Property of Data General Corporation

Sample Packet

```
PKT:    .BLK    ?IBLT           ;Allocate enough space for packet.
                                ;Packet length = ?IBLT.
        .LOC    PKT+?ICH        ;Channel number.
        .WORD   0               ;The OS returns the channel number.

        .LOC    PKT+?ISTI       ;?OPEN options.
        ?ICRF!?RTFX!?OFCR!?OFCE!?OFOT   ;Change record format to that
                                        ;specified in record format
                                        ;field (?ICRF), which is
                                        ;fixed-length (?RTFX),
                                        ;delete and then recreate the
                                        ;file (?OFCR!?OFCE), and open
                                        ;the file for output (?OFOT).

        .LOC    PKT+?ISTO       ;Specify file type.  The OS assumes
        .WORD   0               ;that it is a user data file
                                ;(?FUDF) by default.

        .LOC    PKT+?IMRS       ;Physical block size is
        .WORD   -1              ;2 Kbytes (-1 is the default).

        .LOC    PKT+?IBAD       ;Specify byte pointer to record I/O
                                ;buffer.
        .DWORD  BUF*2           ;Byte pointer to BUF.

        .LOC    PKT+?IRES       ;Reserved.
        .WORD   0               ;You must set this value to 0.

        .LOC    PKT+?IRCL       ;Specify record length. (?READ and
        .WORD   33.             ;?WRITE use this value.) Maximum
                                ;record length is 33-byte fixed
                                ;records.

        .LOC    PKT+?IRLR       ;Record length.  (?READ and ?WRITE
        .WORD   0               ;use this value.)  The OS returns
                                ;this value.

        .LOC    PKT+?IFNP       ;Specify byte pointer to pathname.
        .DWORD  PTH*2           ;Byte pointer to PTH.

        .LOC    PKT+?IDEL       ;Specify delimiter table address.
        .DWORD  -1              ;Use default delimiters: NEW LINE,
                                ;form feed, and carriage return
                                ;(-1 is the default).

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

Notes

- See the descriptions of ?OPEN and ?CLOSE in this chapter.

Licensed Material - Property of Data General Corporation

?REC

Receives an intertask message.

?REC
error return
normal return

Input

AC0	Address of the mailbox that is to receive the intertask message
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Intertask message
AC2	Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?REC is one of several system calls for intertask communications. (The others are ?RECNW, ?XMT, ?XMTW, ?SIGWT, ?WTSIG, and ?SIGNL.) Generally, you use intertask communications to synchronize tasks and to manage critical regions (critical regions are areas of code which need to be protected from being executed by multiple tasks at a given time.)

Because you can use the intertask message facility to pass addresses, the mailboxes for 32-bit tasks must be 32 bits wide, and those for 16-bit tasks must be 16 bits wide.

What It Does

?REC loads AC1 with a message sent by another task, and then clears the contents of the mailbox (that is, sets the mailbox contents to 0). The operating system then readies the receiving task, provided it was not suspended in the meantime by a ?PRSUS or an ?IDSUS call.

If the receiving task issues ?REC before the sending task transmits the message, via ?XMTW or ?XMT, the operating system suspends the ?REC caller until the transmission occurs.

If more than one task issues ?REC against the same mailbox, the highest priority task receives the message, unless the sending task specified that the message should be broadcast to all potential receivers. If the message is not broadcast and all receiving tasks have the same priority, the operating system sends the message to the highest priority receiver and suspends the other tasks that issued subsequent ?RECs against the mailbox.

Notes

- See the descriptions of ?PRSUS, ?RECNW, ?IDSUS, ?XMTW, and ?XMT in this chapter.

Licensed Material - Property of Data General Corporation

?RECNW

Receives an intertask message without waiting.

?RECNW
error return
normal return

Input

AC0	Address of the mailbox that is to receive the intertask message
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Intertask message
AC2	Undefined

Error Codes in AC0

ERNMW No message waiting (There is no outstanding message.)

Why Use It?

Like ?REC, ?RECNW allows a task to receive a message from another task. You would choose ?RECNW over ?REC to avoid suspending the receiving task if there is a delay between the receive and the complementary transmit system call.

What It Does

?RECNW loads AC1 with a message sent by another task, and then clears the contents of the mailbox. The operating system then readies the receiving task, provided it was not suspended in the meantime by an ?IDSUS or a ?PRSUS call.

?RECNW, unlike ?REC, does not suspend the calling task if the mailbox is empty. Instead, ?RECNW fails and returns an error code in AC0. ?RECNW maintains the calling task in the ready state, regardless of the timing of the transmit and receive sequence. This means that a receiving task can issue ?RECNW before the sending task transmits the message (?XMT or ?XMTW) without risking suspension. If the sending task becomes suspended on an ?XMTW to the mailbox, a complementary ?RECNW causes rescheduling.

Notes

- See the descriptions of ?PRSUS, ?RECNW, ?IDSUS, ?XMTW, and ?XMT in this chapter.

?RECREATE

Recreates a file.

?RECREATE
error return
normal return

Input

AC0	Byte pointer to the target file's pathname
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERDID	Directory delete error
ERDIU	Directory in use
ERFAD	File access denied
ERFDE	File does not exist
ERMPR	System call parameter address error
ERPRM	Cannot delete permanent file
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?RECREATE combines the functions of ?DELETE and ?CREATE in that it allows you to delete all user data in a file, but maintain its characteristics. ?RECREATE is useful for releasing a file's disk blocks.

What It Does

?RECREATE deletes the target file or directory, and creates it again with the same ACL, filename, file type, creation date, and user data area (UDA) information, but without data. In short, ?RECREATE gives you a file with the same characteristics as your original file, but with a length of 0.

If the target file is open when you issue ?RECREATE, the operating system maintains all the data until you issue the last ?CLOSE. However, the data is visible only to the processes that opened the file before you issued ?RECREATE.

You cannot issue ?RECREATE against a file or directory with the PERMANENCE attribute or a directory that is currently in use (that is, your working directory or one cited in a search list).

Notes

- See the descriptions of ?CLOSE, ?CREATE, and ?DELETE in this chapter.

?REGISTER — Creates an entry in the Public Name Registry.

AOS/DVS only

?REGISTER *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?REGISTER packet

Error Codes in AC0

ERNAE	Name already exists
EROAR	Object already registered
ERPKT	Invalid packet identifier

File system errors

Why Use It?

Use this system call to create an entry for the specified resource in the Public Name Registry.

For example, a server that wishes to respond to requests from customers in a distributed environment could identify itself by using ?REGISTER. This makes its well-known IPC port name globally visible. Potential customers could then issue ?ILKUP calls against this well-known public name. Such a server would delete (?DEREGISTER) the port name as part of its normal termination processing.

What It Does

A resource becomes globally visible to users on all nodes in an AOS/DVS community as a result of its (or a containing directory's) being identified to the Public Name Registry (PBR), which then creates an appropriate entry in the Public Name Registry. The ?REGISTER system call lets you identify resources that will become globally visible. In contrast, the ?DEREGISTER system call lets you remove a currently registered resource from the PBR's registry.

You can only register resources that the file system manages or names. Examples include directories, physical units, queues, and text files.

AOS/DVS treats multiple attempts to ?REGISTER the same object, whether under its own simple name or under a different name, as errors. Similarly, AOS/DVS returns an error if you attempt to register two different resources under the same publicly visible name.

Figure 2-189 shows the structure of ?REGISTER's parameter packet and Table 2-168 describes its contents.

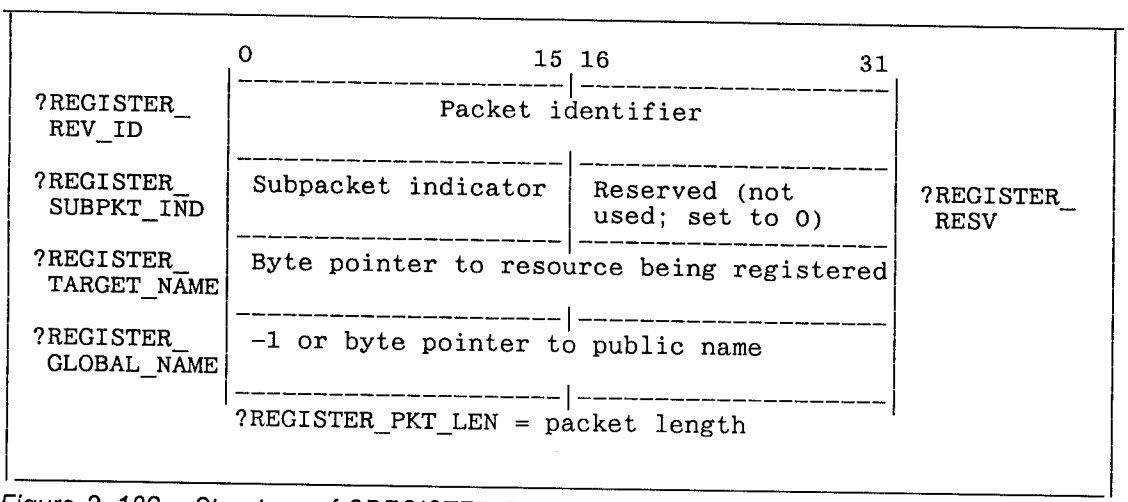


Figure 2-189. Structure of ?REGISTER Packet

Table 2-168. Contents of ?REGISTER Packet

Offset	Contents
?REGISTER_REV_ID (double word)	Packet identifier. Supply ?REGISTER_PKTID.
?REGISTER_ SUBPKT_IND	Subpacket indicator. Set this word to zero because there are no subpackets for this call.
?REGISTER_RESV	Reserved. Set this word to zero.
?REGISTER_ TARGET_NAME (double word)	Supply a byte pointer to the pathname of the resource that you want to register.
?REGISTER_ GLOBAL_NAME (double word)	Supply -1 if the public name of the resource is the same as the simple name. Otherwise, supply a byte pointer to the simple public name.

?RELEASE — Releases an initialized logical disk.

?RELEASE
error return
normal return

Input

AC0	Byte pointer to the pathname of the logical disk (LD)
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERIFT	Illegal file type
ERVBP	Invalid byte pointer passed as a system call argument
ERVIU	LD in use, cannot release
ERWAD	Write access denied

Why Use It?

Once a logical disk has been initialized, it remains initialized until you release it. Therefore, you must issue ?RELEASE to reuse the disk drives on which an LD's component volumes are mounted.

What It Does

?RELEASE releases an LD that you previously initialized with ?INIT. You cannot release an LD that is currently in use. Also, the calling process must have Write access to the LD's parent directory.

Notes

- See the description of ?INIT in this chapter.

?RENAME

Renames a file.

?RENAME *[packet address]*
error return
normal return

Input

AC0	One of the following: • Byte pointer to the file's original filename • 0 if a packet address is supplied
AC1	Byte pointer to the new filename
AC2	One of the following: • Reserved (Set to 0 if not supplying a packet) • Address of the ?RENAME packet, unless you specify the address as an argument to ?RENAME

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined or address of ?RENAME packet if not supplying a packet

Error Codes in AC0

ERIFC	Illegal filename character
ERWAD	Write access denied
ERVBP	Invalid byte pointer passed as a system call argument
ERCRR	Cannot rename the root

?RENAME Continued

Why Use It?

You can use ?RENAME to change the name of a file or LD.

What It Does

?RENAME deletes the target file's current name and gives it the name that you specify in AC1. The target file can be specified in one of two ways: either by a byte pointer to the original pathname in AC0, or by using offset ?GCPCN in the ?RENAME packet. The use of a packet is only necessary if you choose to specify the target file's channel number. AC0 must be set to zero to indicate the use of a channel. If a byte pointer is specified in AC0, then the current (original) name may be a full pathname; but the new name MUST be a simple filename. To use ?RENAME the calling process must have Write access to the target file's parent directory, or Owner access to the target file.

If you use ?RENAME to rename an LD, the new name becomes invalid if you ?RELEASE the LD, and then reinitialize it. (The LD assumes its original name in this case.)

Figure 2-190 shows the structure of the ?RENAME packet.

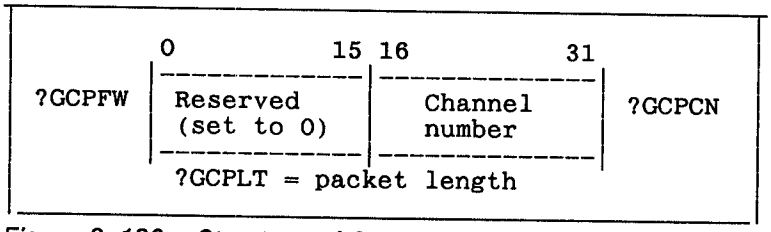


Figure 2-190. Structure of ?RENAME Packet

Licensed Material - Property of Data General Corporation

?RESCHED

Reschedules current time slice. (32–Bit Processes Only)

?RESCHED
error return
normal return

Input

None

Output

None

Error Codes in AC0

ERICM 16–bit program incorrectly attempted to call ?RESCHED

Why Use It?

?RESCHED can be valuable when multiple processes are cooperating. For example, if an executing process reaches a roadblock and must wait for an external event that is triggered by another of the cooperating processes, the executing process can relinquish control of the CPU. Then, the next time the operating system schedules that process, it can see if the roadblock has cleared.

A process that cannot proceed can issue ?RESCHED to allow another process to run.

What It Does

?RESCHED allows the calling process to give up control of the CPU. ?RESCHED causes the operating system to immediately schedule another process for execution.

?RESIGN

Resigns as a server.

?RESIGN
error return
normal return

Input

None

Output

None

Error Codes in AC0

ERNAS Process is not a server

Why Use It?

?RESIGN is one of six disconnect system calls. (The others are ?BRKFL, ?CTERM, ?DCON, ?RETURN, and ?TERM.) Note that ?RESIGN alone does not clear the customer/server entry from the connection table. The operating system clears the entry only when both the customer and the server disconnect.

What It Does

?RESIGN breaks the connection between the calling process and its customers, and sends obituary messages to the customers (unless they set the ?MCOBIT mask in their ?CON system calls). The calling process, which must be a server, issues ?RESIGN to revoke the ?SERVE it issued previously. If the caller is not a server, ?RESIGN fails, and the operating system returns error code ERNAS to AC0.

Notes

- See the descriptions of ?SERVE, ?BRKFL, ?CTERM, ?DCON, ?RETURN, and ?TERM in this chapter.

?RETURN Terminates the calling process and passes the termination message to the father.

?RETURN
error return

Input

AC0 Error code to the CLI (optional)

AC1 Byte pointer to the message (optional)

AC2 Contains the following:

- Bits 0 through 15 (the high-order bits) set to zero
- Bits 16 through 23 (the low-order bits) contain the following flags and data:

  ?RFCF—Message is in CLI format
  ?RFEC—Error code is in AC0
  ?RFWA—Warning (severity bit)
  ?RFER—Error (severity bit)
  ?RFAB—Abort (severity bit)

- Bits 24 through 31 contain the byte length of the message (message length must be between 0 and 255 characters)

Output

None (There is no normal return from ?RETURN.)

Error Codes in AC0

ERMPR System call parameter address error
ERVBP Invalid byte pointer passed as a system call argument

?RETURN Continued

Why Use It?

?RETURN lets you terminate the calling process and pass control back to its father process with an optional termination message from the son. If the father process is the CLI, set the severity bits in AC2. Setting the severity bits in AC2 causes the CLI to flag the reason for the process termination.

What It Does

?RETURN terminates the calling process and returns control to its father.

To send a termination message to the caller's father, load AC1 with a byte pointer to the message, and load Bits 24 through 31 of AC2 with the byte length of the message.

Flag ?RFCF signals that the message is in CLI format. You should set ?RFCF if you intend to send a message, if the father process is EXEC, or if the father process is the CLI. If the father is neither EXEC nor the CLI, you can send the message in any format that the calling process and its father agree upon.

If there is to be no message, set Bits 24 through 31 of AC2 to 0. This directs the operating system to ignore the contents of AC1 and to send a termination code to the father via the IPC facility.

?RETURN also generates an error code in AC0. The severity bits in AC2 direct the father process to handle this error as one of the following types: a warning (?RFWA), an error (?RFER), or an abort condition (?RFAB). If you set one of these bits and the father process is the CLI, the CLI displays

- The message that you specified in AC1 (optional).
- The system error message that corresponds to the error code in AC0, flagged as a WARNING, ERROR, or ABORT.

Note that ?RETURN does not actually affect how the CLI or EXEC handles warnings, errors, and aborts.

To flag a normal return, set AC2 to 0 or specify ?RFCF in AC2.

If a ?RETURN breaks a customer/server connection, the operating system sends both an obituary message and a standard termination message to the father process. Thus, if a server is also the father of its customer, it must issue two ?IREC system calls to receive both messages when the customer issues ?RETURN.

?RINGLD Loads a program file into a specified ring.

?RINGLD
error return
normal return

Input

AC0	Byte pointer to the program file's pathname
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

EREAD	Execute access denied
ERICM	Illegal system command (invalid call for 16-bit processes)
ERMEM	Insufficient memory available
ERPDF	Error in process UST definition
ERPTY	Illegal process type
ERRAL	Ring already loaded
ERRNI	Ring number invalid
ERRTB	Ring too big (The specified program file is too large for the ring.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?RINGLD lets you define code in another user ring. You can use ?RINGLD to load runtime routines into another ring for shared access by processes.

What It Does

?RINGLD loads the program file (.PR file) that you specify in AC0 into Ring 4, 5, or 6. You can issue ?RINGLD from any user rings, including Ring 7, to load another user ring.

The program that you load must contain a properly defined gate array. The gate array identifies the legal entry points for the ring you are loading. In addition, you must access one of the legal gates in the calling process's program.

You can address a program in an inner ring by issuing LCALL instructions (subroutine calls) from tasks within the calling process.

When you link a program that you intend to load into an inner ring, you should set its task count greater than or equal to the task maximum that you set for the calling process. If the new program's task count is greater than that of the calling process, the operating system readjusts it at ?RINGLD time to match the calling process's maximum task count.

The operating system maintains an ECLIPSE MV/8000® stack for every task. When you load a program into an inner ring, the operating system allocates stack space for all its tasks, including those that are currently inactive; the operating system also initializes the stack pointers. All stacks are the same size; that is, they are the size that you specified when you linked the new program.

Notes

- See the descriptions of ?RINGPR and ?RINGST in this chapter.
- Refer to the *Principles of Operation 32-Bit ECLIPSE® Systems* manual for information on gates and gate arrays.

?RNAME Determines whether a pathname contains a reference to a remote host.

?RNAME
error return
normal return

Input

AC0	Byte pointer to pathname
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	One of the following: • 0 if host reference is local • −1 if host reference is remote and host ID is 0 • Host ID in Bits 17 through 31
AC1	Undefined
AC2	Undefined

Error Codes in AC0

File system error codes

Why Use It?

?RNAME allows you to detect remote references in pathnames.

What It Does

Given a pathname, ?RNAME returns the host ID of the host on which the file should exist. Note that this does not mean that the file does exist.

If ?RNAME returns a host ID of −1 in Bits 17 through 31 of AC0, the pathname referenced a remote host with a host ID of 0.

?RNGPR Returns the .PR filename for a ring.

?RNGPR
error return
normal return

Input

AC0 One of the following:

- PID
- Byte pointer to the process name
- For information about self, -1

AC1 One of the following:

- -1 if AC0 contains a byte pointer to the process name
- Any number other than -1 if there is a PID or a -1 in AC0

AC2 Word pointer to the ?RNGPR packet

Output

Length of the string returned (in offset ?RNGLB.)

Error Codes in AC0

ERIRB	Insufficient room in buffer
ERPRH	Process not in hierarchy
ERRNI	Ring number invalid
ERRNL	Ring not loaded
ERVBP	Illegal byte pointer
ERVWP	Illegal word pointer

Why Use It?

You can use ?RNGPR to find out what program the operating system has loaded into a specific ring or whether it has loaded a program at all.

What It Does

?RNGPR allows a process to verify which program has been loaded into a specific ring. If the ring has not already been loaded, the operating system returns error code ERRNL in AC0.

You can request this information for the calling process, for a specified PID, or by specifying a process name. The buffer to which the caller's packet points never exceeds ?MXPL words.

To use ?RNGPR, you must supply a packet that contains a byte pointer to the buffer and the number of the ring you want to check. Figure 2-191 shows the structure and Table 2-169 lists the contents of the ?RNGPR packet.

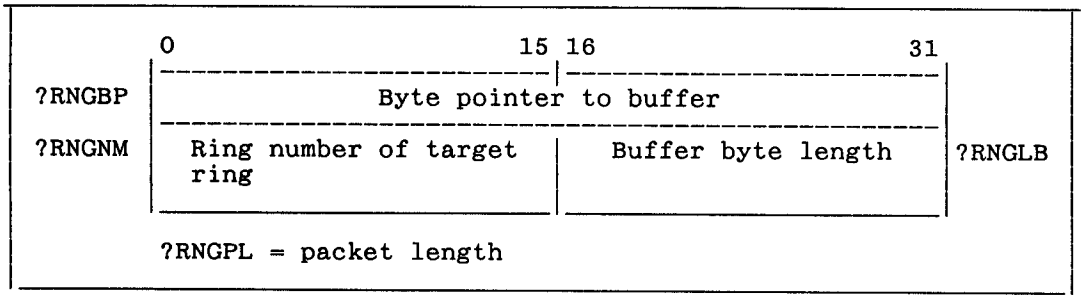


Figure 2-191. Structure of the ?RNGPR Packet

Table 2-169. Contents of the ?RNGPR Packet*

Offset	Contents
?RNGBP (double word)	Byte pointer to buffer.
?RNGNM	Number of ring to check.
?RNGLB	Length of buffer (in bytes).

* There is no default unless otherwise specified.

Notes

- See the descriptions of ?RINGLD and ?RINGST in this chapter.

?RNGST Stops ?RINGLD from loading lower rings.

?RNGST
error return
normal return

Input

AC0	Logical address of the lowest ring that allows loading
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPTY	Illegal process type
ERRNI	Ring number invalid
ERRAL	Ring already loaded

Why Use It?

?RNGST prevents a user's ?RINGLD from circumventing a ring's protection mechanisms.

What It Does

?RNGST lets a process specify the lowest ring that ?RINGLD can load. ?RNGST allows servers to prevent themselves from being circumvented by a user process in Ring 7 that is loading a lower ring. If user rings below the specified address are already loaded, the operating system returns the error code ERRAL to AC0. Also, in that case, AC1 contains the ring number of the ring that failed.

A caller can specify that ?RINGLD cannot load programs into rings that are higher than the caller's own ring. However, in this case, the operating system ignores the fact that the caller's ring is already loaded.

The ring-maximization rules apply to ?RNGST. Therefore, an outer-ring ?RNGST cannot override an inner-ring ?RNGST.

?RNGST does not affect system rings.

Notes

- See the descriptions of ?RINGPR and ?RINGLD in this chapter.

Licensed Material - Property of Data General Corporation

?RPAGE — Releases a shared page.

?RPAGE
error return
normal return

Input

AC0 Contains the following:

- Bit 0 is a flag bit:

  If Bit 0 = 1, flush the contents of the page to disk before releasing it

  If Bit 0 = 0, do not flush the contents of the page to disk before releasing it

- Bits 1 through 31 contain any address in the target page (The OS converts this value to the correct logical page number.)

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

Why Use It?

?RPAGE removes a shared page from your logical address space, but does not necessarily remove it from memory. This eliminates the need for disk I/O, if another process requires the same page.

When you set the flag bit (Bit 0) in AC0, the ?RPAGE simulates a ?FLUSH; that is, the operating system retains the page in memory, but flushes its contents to disk if it is modified. This option guarantees that the shared page is updated immediately.

?RPAGE Continued

What It Does

?RPAGE releases a shared memory page from the caller's logical address space. If no other process is using the target page after the ?RPAGE, the operating system places it on the LRU chain for eventual reuse. (Note that the page remains in the user context; ?RPAGE does not deallocate the page.)

If you set Bit 0 of AC0 and the page has been modified, the operating system flushes the contents of the page to disk. This variation is functionally the same as issuing ?FLUSH followed immediately by a standard ?RPAGE.

Licensed Material - Property of Data General Corporation

?RUNTM Gets runtime statistics on a process.

?RUNTM *[packet address]*
error return
normal return

Input

AC0	One of the following: • PID of the target process • Byte pointer to the name of the target process • −1 to obtain statistics for the calling process
AC1	One of the following: • −1 if AC0 contains a byte pointer • 0 if AC0 contains a PID Otherwise, the OS ignores AC1
AC2	Address of the ?RUNTM packet, unless you specify the address as an argument to ?RUNTM

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPRH	Attempt to access process not in hierarchy
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

?RUNTM Continued

Why Use It?

Like ?PSTAT and ?WHIST, ?RUNTM returns information that you can use to judge a process's performance and its use of system resources.

What It Does

?RUNTM returns runtime statistics for either the calling process or any other process the caller specifies. Before you issue ?RUNTM, load AC0 with the PID of the target process or with a byte pointer to its name. If you specify -1 in AC0, the operating system returns statistics for the calling process.

You must also reserve a packet of ?GRLTH words to receive the ?RUNTM statistics. You can cite the packet address as an argument to ?RUNTM or you can load its address into AC2 before you issue ?RUNTM. Figure 2-192 shows the packet structure.

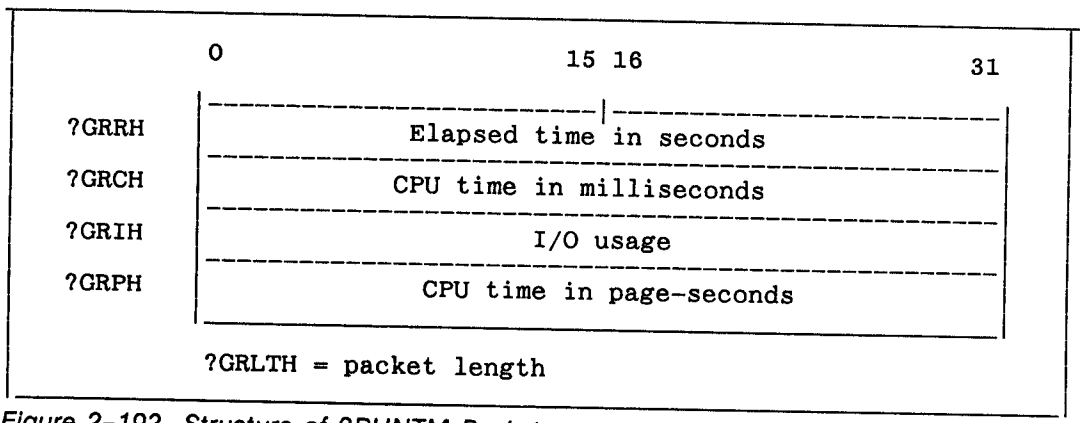


Figure 2-192. Structure of ?RUNTM Packet

As Figure 2-192 shows, the operating system returns the following information to the ?RUNTM packet: the elapsed time of the target process, its CPU time, its page usage over CPU time, and its I/O usage (number of blocks read or written).

Sample Packet

```
PKT:    .BLK    ?GRLTH          ;Allocate enough space for packet.
                                ;Packet length = ?GRLTH.
        .LOC    PKT+?GRRH       ;The OS will return elapsed time
        .DWORD  0               ;(in seconds).
        .LOC    PKT+?GRCH       ;The OS will return CPU time (in
                                ;milliseconds).
        .DWORD  0
        .LOC    PKT+?GRIH       ;The OS will return I/O usage
        .DWORD  0               ;information.
        .LOC    PKT+?GRPH       ;The OS will return page usage
        .DWORD  0               ;information.
        .LOC    PKT+?GRLTH      ;End of packet.
```

Notes

- See the descriptions of ?PSTAT and ?WHIST in this chapter.

?SACL — Sets a new access control list (ACL).

?SACL *[packet address]*
error return
normal return

Input

AC0 One of the following:

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

AC1 One of the following:

- Byte pointer to the new ACL
- −1 to delete the current ACL

AC2 One of the following:

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

Output

AC0 Unchanged

AC1 Undefined

AC2 Undefined or address of ?SACL packet

Error Codes in AC0

ERACL	Illegal ACL
ERVBP	Invalid byte pointer passed as a system call argument
ERWAD	Write access denied

Licensed Material - Property of Data General Corporation

?SACL allows you to alter or delete a file's current access control list (ACL). You can use ?SACL to change the ACL you set when you created the file.

The operating system reverts to the default access privileges when you ?DELETE, and then ?CREATE a file.

To find out what the ACL is for a particular file or directory before you issue ?SACL, issue ?GACL.

What It Does

?SACL replaces the target file's or directory's ACL with the ACL that you specify in AC1. If there is no new ACL, ?SACL deletes the existing ACL.

The calling process must have Write access to the target's parent directory or Owner access to the target itself.

You can specify the target file or directory in one of two ways: either by a byte pointer to the file or directory's pathname in AC0, or by using offset ?GCPCN in the ?SACL packet. The use of the packet is only necessary if you choose to specify the target file or directory's channel number. AC0 must be set to zero to indicate the use of a packet.

Before you issue ?SACL, define the new ACL in your address space and load AC1 with a byte pointer to its address. Structure the ACL specification as follows:

username<0><access privs>[username<0><access privs>...]<0>

For an example of 9 bytes:

OP<0><?FAC0+?FACW+?FACA+?FACR+?FACE>SK<0><?FACR+?FACE><0>

where

- access privs is one or more of the following access types:

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

The brackets ([ ]) in the format mean that you can repeat the username<0><access privs> entry. However, be sure to type the ACL specifications on one line. If you use a carriage return or NEW LINE character, the operating system interprets that character as part of the ACL. Do not insert any spaces between the ACL mnemonics, and do terminate the ACL specification with a null byte.

?SACL Continued

To give a series of access privileges to all users, use the +<0> template before the ACL specification. For example, the specification +<0><?FACR+?FACE><0> gives all users Read and Execute access to the file. The symbol ?MXACL represents the maximum length for an ACL specification.

For another example, the CLI command

```
ACL    FOO SAM,OAR    LYNN,RE
```

sets the ACL of file FOO. You can also do this with the ?SACL system call. One step is to create a buffer containing the following 12. bytes in its leftmost bytes.

SAM<0><?FACO+?FACA+?FACR>LYNN<0><?FACR+?FACE><0>Each of the symbols <?FACO+?FACA+?FACR> and <?FACR+?FACE> requires 1 byte.

Figure 2-193 shows the structure of the ?SACL packet.

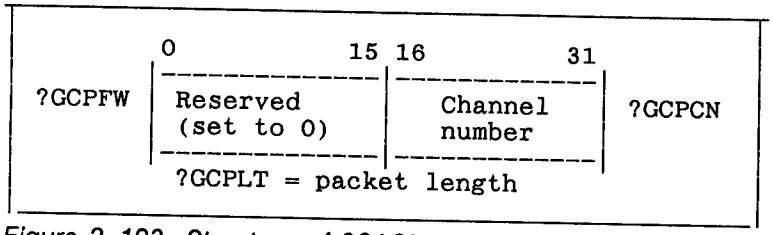


Figure 2-193. Structure of ?SACL Packet

Notes

- See the descriptions of ?CREATE, ?DELETE, and ?GACL in this chapter.

Licensed Material - Property of Data General Corporation

?SATR — Sets or removes the permanent attribute for a file or directory.

?SATR *[packet address]*
error return
normal return

Input

AC0 — One of the following:

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

AC1 — Bit 31 is a flag bit:

- Bit 31 = 1 to set the permanent attribute
- Bit 31 = 0 to remove the PERMANENCE attribute

AC2 — One of the following:

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

Output

AC0 — Unchanged

AC1 — Unchanged

AC2 — Undefined or address of ?SATR packet

Error Codes in AC0

ERIFT	Illegal file type
ERVBP	Invalid byte pointer passed as a system call argument
ERWAD	Write access denied

Why Use It?

You can protect a file from being deleted by any process, regardless of the process's access privileges, by assigning that file the permanent attribute. If you assign the PERMANENCE attribute to a file, you should assign that attribute to its parent directory as well. Otherwise, a process can delete the file indirectly, simply by deleting the parent directory.

To find out if a file has the PERMANENCE attribute, issue ?FSTAT and examine the ?SSTS offset in the ?FSTAT return packet.

What It Does

?SATR either assigns or removes the PERMANENCE attribute for a file or directory, depending on your input to AC1. You cannot delete any entries that have the PERMANENCE attribute. (The PERMANENCE attribute is the only file attribute that is currently defined.)

To use ?SATR, the calling process must have Owner access to the target entry or Write access to the entry's parent directory.

The target file can be specified in one of two ways: either by a byte pointer to the entry's pathname in AC0, or by using offset ?GCPCN in the ?SATR packet. The use of the packet is only necessary if you choose to specify the target entry's channel number. AC0 must be set to zero to indicate the use of a packet.

Figure 2-194 shows the structure of the ?SATR packet.

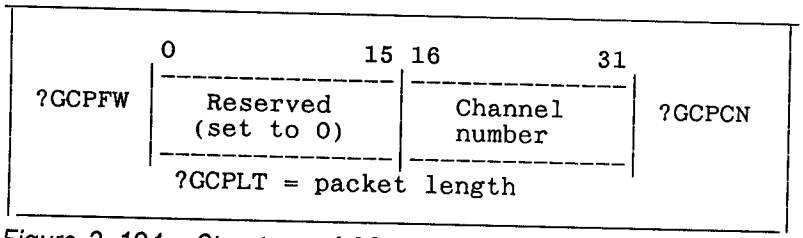


Figure 2-194. Structure of ?SATR Packet

Notes

- See the description of ?FSTAT in this chapter.

Licensed Material - Property of Data General Corporation

?SBIAS

Sets the bias factors.

?SBIAS
error return
normal return

Input

AC0	Contains the following: • Maximum bias value in Bits 16 through 23 • Minimum bias value in Bits 24 through 31
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERBIF	Illegal bias factor
ERPRV	Caller not privileged for this action (The caller is not PID 2.)

Why Use It?

By setting maximum and minimum bias factors, you can influence the operating system's normal bias towards interactive processes (processes that interact with the operating system through a console). The system manager or operator usually sets the bias factors during the system-generation procedure.

To find out what current bias factor values have already been set for your system, issue ?GBIAS.

?SBIAS Continued

What It Does

?SBIAS sets the maximum and minimum bias factors. Only the operator process (PID 2), or a process that has System Manager privilege, can issue ?SBIAS.

The minimum bias factor represents the fewest number of noninteractive processes the operating system will try to maintain in memory at one time. You must set a minimum bias factor; either with ?SBIAS or during the system-generation procedure. In general, the minimum bias factor should be low to avoid unfair bias toward noninteractive processes.

The maximum bias factor represents the maximum number of noninteractive processes the operating system will allow in memory before favoring blocked interactive processes. The maximum bias factor is optional.

If you do set the maximum bias factor, note that it is not absolute. Even after reaching the maximum bias value, the operating system can run another noninteractive processs if it can do so without pre-empting another process.

Notes

- See the description of ?GBIAS in this chapter.
- Refer to your operating system's "How to Generate and Run" manual for information on the system-generation procedure.

Licensed Material - Property of Data General Corporation

?SCHR Sets a character device's characteristics.

?SCHR
error return
normal return

Input

AC0	One of the following: • Byte pointer to the device name • The number of the channel on which the device is open
AC1	Bits 0 and 1 are flag bits: • Bit 0 = 0 if AC0 contains a byte pointer • Bit 0 = 1 if AC0 contains the channel number • Bit 1 = 0 to change the device characteristics • Bit 1 = 1 to define default characteristics
AC2	Address of the three-word device characteristics specification

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERICN	Illegal channel
ERIFT	Illegal file type
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

When a character device is in text mode, the operating system uses the characteristics of the device to determine how to interpret the output from that device. Although you usually set device characteristics during the system-generation procedure, you can use ?SCHR to reset certain characteristics for the duration of the calling process.

What It Does

?SCHR allows you to set a character device's characteristics. To do this, select certain options within the three device characteristics words. ?SCHR allows

- Any process to change the characteristics set for a character device to which the caller has access.
- The operator process (PID 2) to set new default characteristics that override the defaults that were chosen during the system-generation procedure.

Only the operator process (PID 2) can define new default characteristics for a character device.

Before you issue ?SCHR, load AC0 with the channel number of the target device or with a byte pointer to the device name. Then, load AC2 with the address of the "new" device characteristics words, which you must supply elsewhere in your program. (See Table 2-170 for a complete list of the device characteristics words.)

If AC0 contains a byte pointer, set Bit 0 of AC1 to 0; if AC0 contains the device's channel number, set Bit 0 of AC1 to 1. Bit 1 of AC1 defines whether this ?SCHR system call will change only some of the characteristics or will set new default characteristics.

You can also set what are known as "extended" characteristics for a character device. The flags in the two words following the three which ?SCHR uses control XON/XOFF data flow over console lines and some of the operating characteristics of Intelligent Asynchronous Controllers (IACs).

To read the characteristics of a character device, use the ?GCHR and ?GECHR system calls.

Note that only the operator process (PID 2) can use ?SCHR to create new default characteristics for a character device. In addition, the operator process can use ?SCHR to set the two modem-related characteristics, modem control (?CMOD) and monitor ring indicator (?CMRI), only if its own default characteristics allow it to do so.

Notes

- See ?SECHR in this chapter for setting extended characteristics .
- Refer to your operating system's "How to Generate and Run" manual for information on setting operator process defaults.

Licensed Material - Property of Data General Corporation

Table 2-170. Character Device Characteristics Words

Word	Characteristic	Meaning
0	?CST	Simulates 8-column tabs.
	?CSFF	Simulates form feeds.
	?CEPI	Requires even parity on input.
	?C8BT	Allows 8 data bits per character. You must set this characteristic when you need all 8 bits of each byte when you perform a binary read.
	?CSPO	Sets even parity on output.
	?CRAF	Sends 17. rubout characters after each form feed, unless the form feed is simulated.
	?CRAT	Sends two rubout characters after each tab, unless the tabs are simulated.
	?CRAC	Sends two rubout characters after each carriage return or NEW LINE.
	?CNAS	Non-ANSI-standard device.
	?COTT	Converts octal 175 and octal 176 to octal 33 (ESC) on output.
	?CEOL	No automatic carriage return or line feed at end of each line. (If ?CWRP is off and ?CEOL is on, then the output truncates at the line length defined by the device characteristics.)
	?CUCO	Uppercase output only.
	?CMRI	Monitors rings indicator on modem control line.
	?CFF	Form feed on open.
	?CEB0+?CEB1	Echo mode (one of the following). EB0 = 0 No echoing. EB1 = 0 ?CEOS--Echo exactly as input. EB1 = 1 ?CEOC--Echo control characters with uparrows (e.g., ^A).

(Continues)

Table 2-170. Character Device Characteristics Words

Word	Characteristic	Meaning
1	?CULC	Lowercase or uppercase input only (not both).
	?CPM	Page mode.
	?CNRM	Disables message receipt (accept no ?SEND system calls).
	?CMOD	Modem control.
	?CDT0+?CDT3	Device type (one of the following) ?TTY 4010A hard-copy console ?CRT1 4010I display console ?CRT2 6012 display console ?CRT3 605x display console ?CRT6 6130 display console ?CRT7 - ?CRT15 user defined devices
	?CTO	Enables device time-outs.
	?CTSP	Retains trailing blanks (card readers only).
	?CPBN	Right-justifies each column to 16-bits (card readers only).
	?CESC	Each ESC character generates a CTRL-C CTRL-A interrupt.
	?CWRP	(Hardware) Wraparound for lines longer than the characters per line defined in the last characteristic word.
	?CFKT	Uses function keys as input delimiters (device types ?CRT3 and ?CRT6 only).
	?CNNL	Does not append NEW LINE characters.
2	Left byte	Number of lines per page (for Page mode devices).
	Right byte	Characters per line (CPL).

(Concluded)

Licensed Material - Property of Data General Corporation

?SCLOSE

Closes a file previously opened for shared access.

```
?SCLOSE
error return
normal return
```

Input

AC0 Reserved (Set to 0.)

AC1 Contains the following:

- Bit 0 is a flag bit:

  Bit 0 = 1 to release the shared page(s) from the caller's logical address space

  Bit 0 = 0 to retain the shared page(s) from the caller's logical address space

- Bits 2 through 31 contain the file's channel number (returned by the OS as output to ?SOPEN)

AC2 Reserved (Set to 0.)

Output

AC0 Undefined

AC1 Unchanged

AC2 Undefined

Error Codes in AC0

ERCIU	Channel in use
ERFNO	Channel not open

Why Use It?

You must issue ?SCLOSE to close a shared file. However, you cannot use ?SCLOSE to close a file that you opened with ?OPEN or ?GOPEN.

After the operating system executes ?SCLOSE, it updates the file's status information and frees the channel so that it can be used again.

?SCLOSE Continued

What It Does

?SCLOSE closes a file that ?SOPEN previously opened for shared access.

You can direct the operating system to release any pages of the target page that are still in the caller's logical address space by setting Bit 0 of AC1. If you do not set Bit 0 and pages of the file are still in the address space, the operating system takes the error return from ?SCLOSE and returns error code ERCIU in AC0.

Notes

- See the descriptions of ?SOPEN, ?OPEN, and ?GOPEN in this chapter.

Licensed Material - Property of Data General Corporation

?SDAY

Sets the system calendar.

?SDAY
error return
normal return

Input

AC0	Day from 1 through 31
AC1	Month from 1 through 12
AC2	Year minus 1900 (The result is expressed in octal.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERTIM	Illegal time argument (The day, month, or year value is outside the legal range.)
ERPRV	Caller not privileged for this action (A process other than the operator process attempted to set the day, month, or year.)

Why Use It?

?SDAY lets you (PID 2) adjust the operating system's internal calendar while your program is executing. Usually, the system manager or operator sets the calendar during the system-generation procedure.

What It Does

?SDAY sets the system calendar to the day, month, and year you specify in AC0, AC1, and AC2, respectively. Only the operator process (PID 2), or a process with System Manager privilege, can issue ?SDAY. The operating system automatically increments the date when the time of day passes 23 hours, 59 minutes, 59 seconds.

?SDBL

Disables a BSC line.

?SDBL
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Channel number assigned to the line
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERDSL	Device associated with the channel number is not a synchronous line

Why Use It?

You can use ?SDBL to disable a BSC line previously enabled with ?SEBL.

What It Does

?SDBL disables the BSC line associated with the channel number that you specify in AC1. When you issue ?SDBL, the operating system breaks the association between the line and its channel, and the association between the line and the process that enabled it.

Notes

- See the description of ?SEBL in this chapter.

Licensed Material - Property of Data General Corporation

?SDLM

Sets a delimiter table.

?SDLM
error return
normal return

Input

AC0	One of the following: • Byte pointer to the device name • Channel number of the file
AC1	Flag bits: • ?SDCN if AC0 contains a channel number • ?SDDN if AC0 contains a byte pointer to a device name • ?SDTO to output a delimiter table • ?SDTI to input a delimiter table • ?SDTP priority input delimiter table
AC2	Word address of the 16-word delimiter table

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERICN	Illegal channel
ERIFT	Illegal file type
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

?SDLM Continued

Why Use It?

?SDLM allows you to override the default delimiters for data-sensitive records by defining a new delimiter table. In addition, ?SDLM overrides the delimiter table that you defined in the ?OPEN packet for the file. Thus, you can use ?SDLM to redefine the delimiter table you set up at ?OPEN time.

What It Does

?SDLM sets a delimiter table for a file with the data-sensitive records or for a character device, while the file or device is open. ?SDLM is an alternative to the delimiter table feature in the ?OPEN packet.

Before you issue ?SDLM, set up a delimiter table of 16 consecutive words in your logical address space and load the word address of the table into AC2. (For information on the structure of delimiter tables, see the description of the ?OPEN system call.)

If you load −1 into AC2, the operating system reverts to the default delimiters for data-sensitive files; that is, NEW LINE, carriage return, null, and form feed. If you load 0 into AC2, the null character (0) is the only valid delimiter.

If you issue ?SDLM for a character device, such as a console, you can define one of three types of delimiter tables; your input to AC1 defines the type. For an output delimiter table (a table that defines delimiters for output to the device), specify bit ?SDTO in AC1. For an input delimiter table (a table that defines delimiters for input to the device), specify bit ?SDTI in AC1. If you have opened the device with the priority read option, specify ?SDTP in AC1 to designate the table as a priority input delimiter table.

If you define a delimiter table for a file in the ?OPEN packet, and then issue ?SDLM, the new delimiter table overrides the delimiter table that you defined in the ?OPEN packet.

Should you want to save an earlier delimiter table before creating a new one with ?SDLM, the ?GDLM system call returns the delimiter table.

Notes

- See the description of ?GDLM and ?OPEN in this chapter.

?SDPOL Defines a polling list or a poll–address/ select–address pair.

?SDPOL
error return
normal return

Input

AC0 One of the following values in Bits 16 through 23:

- Length of each poll address entry (including device address characters), if the caller is a control station
- Length of the poll address, if the caller is a tributary (the select address is the same length)

One of the following values in Bits 24 through 31:

- Number of relative consoles (tributaries) in the polling list, if the caller is a control station
- 1 if the caller is a tributary

AC1 Channel number assigned to the line

AC2 Byte pointer to the polling list

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERIMM	Not enough memory for poll/select list
ERLNA	I/O request for disabled line
ERLNM	Line not multipoint
ERNSL	Attempt to enable nonsynchronous line

?SDPOL Continued

Why Use It?

A control station must define a polling list with ?SDPOL before it can perform polling or selecting. Similarly, a tributary station must define its poll and select addresses with ?SDPOL before its control station can poll or select it.

What It Does

?SDPOL performs the following functions:

- If the caller is a control station on a multipoint line, ?SDPOL defines a polling list.
- If the caller is a tributary on a multipoint line, ?SDPOL defines the caller's poll address and select address.

Note that only stations on a multipoint bisynchronous communications (BSC) line can issue ?SDPOL.

The operating system uses the addresses that ?SDPOL defines during receiving initial (?SRCV) general polling sequences. If your computer is a control station, you must issue ?SDPOL before you can perform polling or selecting. If your computer is a tributary, you must issue ?SDPOL to define your poll and select addresses.

Defining a Polling List

If you use ?SDPOL from a control station, construct the polling list in contiguous locations in your logical address space before you issue ?SDPOL. The entries in the list can be of any byte length, provided each entry is the same length. Figure 2-195 shows a sample polling list, where each entry is 4 bytes long.

Before you issue ?SDPOL, load AC0 with the byte length of the entries (Bits 16 through 23) and the number of entries (Bits 24 through 31). The operating system uses these two values to determine the length of the polling list and to assign relative console numbers to the tributaries.

Defining a Poll Address and Select Address for a Tributary

The procedure for defining a single poll address and select address entry for a tributary is similar to the procedure for defining a polling list. First, define the poll and select addresses in contiguous locations in your logical address space. Then, before you issue ?SDPOL, load Bits 16 through 23 of AC0 with the byte length of the poll or select address. (Each address must be the same byte length.) Load Bits 24 through 31 of AC0 with 1. Figure 2-196 shows a sample poll address and select address entry, where each address is 2 bytes.

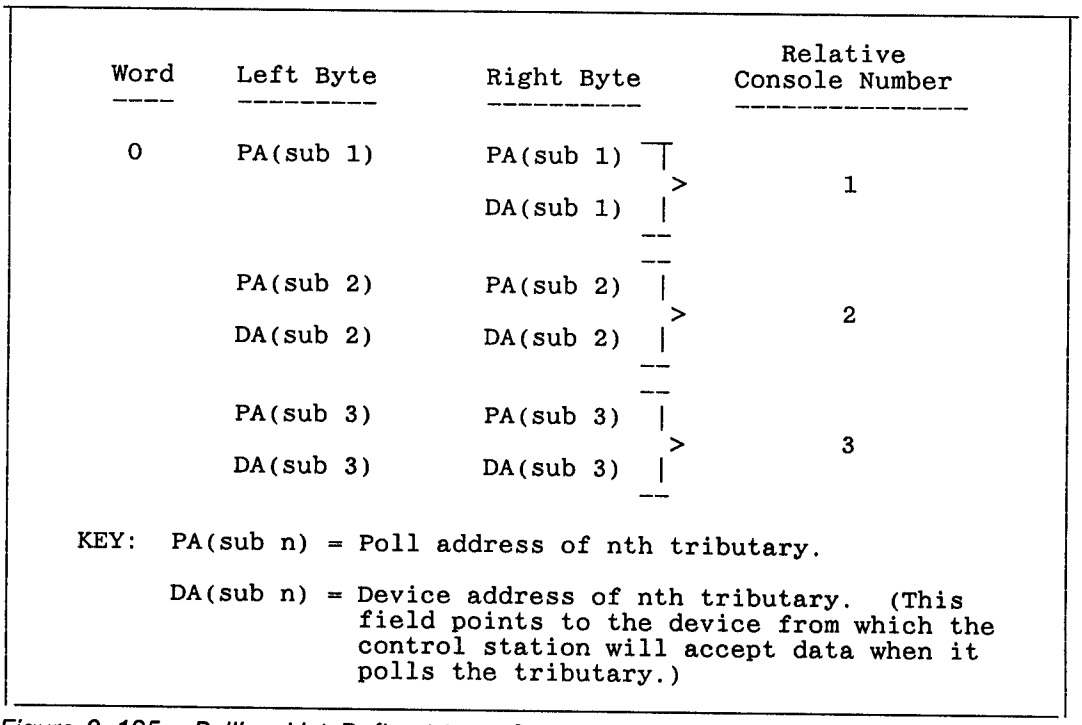


Figure 2–195. Polling List Defined by a Control Station

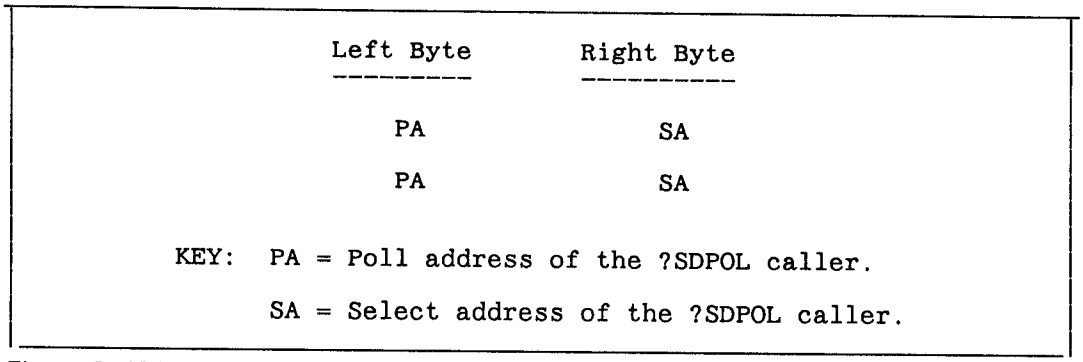


Figure 2–196. Poll and Select Addresses Defined by a Tributary

Notes

- See the description of ?SRCV in this chapter for more information on polling.

?SDRT/?SERT Disables/re-enables a relative console.

?SDRT
error return
normal return

?SERT
error return
normal return

Input

AC0 One of the following values in Bits 16 through 23:

- Length of each poll address entry (including device address characters), if the caller is a control station
- Length of the poll address, if the caller is a tributary (the select address is the same length)

One of the following values in Bits 24 through 31:

- Number of relative consoles (tributaries) in the polling list, if the caller is a control station
- 1 if the caller is a tributary

AC1 Channel number assigned to the line

AC2 Byte pointer to the polling list

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERIRT	Illegal relative console number
ERLNA	I/O request for disabled line (?SERT)
ERLNM	Line not multipoint
ERNPL	Polling list not defined
ERNSL	Attempt to enable nonsynchronous line (?SERT)

Why Use It?

When a control station defines a polling list with ?SDPOL, the operating system automatically enables all the relative consoles on the list. In effect, ?SDRT "removes" a relative console from the polling list by directing the operating system to ignore that console during the next polling sequence. In general, you disable a tributary if it repeatedly fails to respond to a poll or to a select sequence.

?SERT allows the control station to "undo" a previous ?SDRT; that is, to re-enable a previously disabled relative console.

Disabling a relative console does not affect the corresponding tributary station. Instead, the control station ignores the tributary station until you subsequently re-enable the relative console or define a new polling list.

What It Does

?SDRT and ?SERT respectively disable and re-enable a relative console for subsequent polling or selecting. Only a control station on a multipoint line can issue ?SDRT and ?SERT.

Notes

- See the description of ?SCRV in this chapter for more information on polling.

?SEBL

Enables a BSC line.

?SEBL
error return
normal return

Input

AC0	Byte pointer to the line's device name
AC1	Reserved (Set to 0.)
AC2	Address of the ?SEBL packet

Output

AC0	Unchanged
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERDCU	DCU inoperative (can't be initialized)
EREPE	Line already enabled on ?SEBL call
ERFCT	Failure to connect
ERINE	Initialization parameter error (You passed an invalid packet address.)
ERNSL	Attempt to enable nonsynchronous line

Why Use It?

You must use ?SEBL to enable a BSC line before you can use it.

If your computer is a control station on a multipoint line, ?SEBL enables the common line for you and your tributaries to use.

What It Does

?SEBL enables a bisynchronous (BSC) line and associates it with the calling process. ?SEBL also directs the operating system to acquire resources for the line and to perform hardware–related initialization.

Before you issue ?SEBL, set up the ?SEBL packet in your logical address space, load AC2 with the packet address, and load AC0 with a device name for the line.

Licensed Material - Property of Data General Corporation

The device name for a BSC line is

@SLNx

where

- @SLN is the BSC designator.
- x is the line number.

Figure 2–197 shows the structure of the ?SEBL packet, and Table 2–171 describes the contents of each offset.

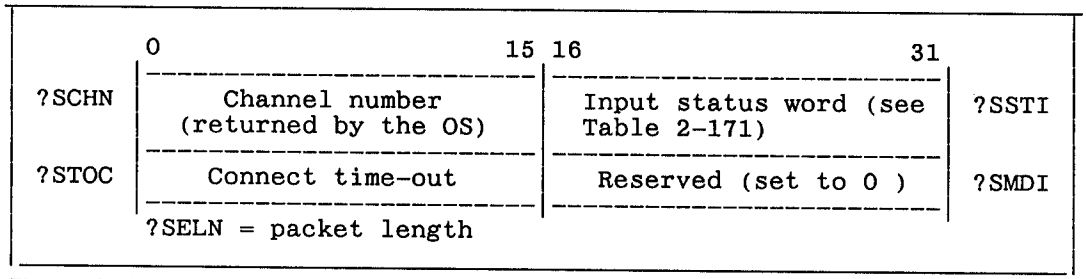


Figure 2–197. Structure of ?SEBL Packet

The Input Status Word

Offset ?SSTI contains specifications for the configuration, status, character set, parity, and protocol of the line you enable.

If you enable a point–to–point line (you selected ?SDPP as the line type), set the line mode to ?SDPR if your computer is the primary station. If your computer is the secondary station, set the line mode to ?SDSC. However, if you enable a multipoint line, set the line mode to ?SDPR if your computer is the control station and to ?SDSC if it is a tributary.

Table 2-171. Contents of ?SEBL Packet*

Offset	Contents
?SCHN	Channel number (returned by the OS).
?SSTI	Input status word. Line type. ?SDPP--Point-to-point. ?SDMD--Multipoint. DEFAULT = ?SDPP. Line mode. ?SDSC--secondary (point-to-point), tributary (multipoint). ?SDPR--primary (point-to-point), control station (multipoint). DEFAULT = ?SDSC. Line code. ?SEBC--EBCDIC. ?SASC--ASCII. DEFAULT = ?SEBC. Block Check. ?SCRC--CRC16. ?SCIT--CCITT16. ?SLRC--LRC. DEFAULT = ?SCRC. Parity. ?SNPR--No parity. ?SOPR--Odd parity. ?SEPR--Even parity. DEFAULT = ?SNPR.

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

Table 2-171. Contents of ?SEBL Packet*

Offset	Contents
?SSTI	Protocol. ?SBSC--Normal BSC protocol. ?SHSP--HASP multileaving. DEFAULT = ?SBSC.
?STOC	Connect time-out. -1 30 seconds. 0 No time-out. 16383 Maximum time-out. DEFAULT = -1 (30 seconds).
?SMDI	Reserved. (Set to 0.)

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

The line code tells the operating system which character set, EBCDIC or ASCII, to use for the data-link control characters. The default character set is EBCDIC. Note that the operating system does not translate text or header information into the character set you specify here. You must translate the outgoing data, if necessary. However, the operating system returns incoming data to you in the character set in which it was received.

The block check field lets you select the block check scheme you want to use for the transmission. Block checking is the method that the operating system uses to validate the entire text block, according to the BSC protocol. (See Table 2-172 for a description of the block check data-link character.) The sending and receiving stations on a BSC line must use the same block check method. Otherwise, the operating system cannot verify the contents of the transmission blocks.

If your line code is EBCDIC (you selected ?SEBC or did not specify the line code), choose either ?SCRC or ?SCIT as the block check type. Both masks correspond to cyclical redundancy checking. (See Table 2-172.) ?SCRC corresponds to the CRC16 method and ?SCIT corresponds to the CCITT16 method.

Table 2-172. BSC Protocol Data-Link Control Characters (DLCC)

Character	Description
ACK0 ACK1	Affirmative Acknowledgment. Positive replies, sent in alternating sequence, to indicate that the receiver has accepted the previous block without error, and is ready to accept the next block of the transmission. ACK0 is also a positive response to a line bid (ENQ) for a point-to-point line and to a selection sequence on a multipoint line.
BCC	Block Check Character. A value generated by the transmitting station and sent with each data block to validate the block's contents. The receiving station generates its own BCC. If the two values match, the block is accepted as error-free. A BCC follows every ITB, ETB, and ETX character. If you transmit in the ASCII code set, the BCC is a longitudinal redundancy check (LRC). For the EBCDIC code set, the BCC is a cyclical redundancy check (CRC).
DLE	Data-Link Escape. The first character in a two-character sequence used to signal the beginning or end of Transparent Text mode. The sequence DLE STX signals the beginning of Transparent Text mode. The sequence DLE ETB or DLE ETX signals the end of Transparent Text mode.
DLE EOT	Data-Link Escape, End-of-Transmission. Signals a line disconnect for a switched line. The sending or receiving station usually transmits this sequence when all message exchanges are finished.

(Continues)

Table 2-172. BSC Protocol Data-Link Control Characters (DLCC)

Character	Description
ENQ	Enquiry. Sent by a station on a point-to-point line to bid for the line (for transmission of data). Sent by the control station on a multipoint line to signal the end of a polling or selecting sequence. A transmitting station can also send ENQ to ask the receiver to repeat a response if the original response was garbled or not received when expected.
EOT	End-of-Transmission. Signals the end of a message transmission (consisting of one or more separately transmitted data blocks), and resets the receiving station. On a multipoint line, a polled station sends an EOT to indicate that it has nothing to send back to the control station. EOT can also serve as an abort signal to indicate a system or transmission malfunction.
ETB	End-of-Transmission Block.
ETX	End-of-Text. Signal the end of a data block that began with an SOH or an STX. Both the ETB and the ETX characters reverse the direction of the transmission. When a station receives an ETB or an ETX, it replies with a control character that indicates its status (that is, ACK0, ACK1, NAK, WACK, or RVI). An ETB terminates every text block except the last. An ETX implies an end-of-file condition; thus, it terminates the last block of text in a message.

(Continued)

Table 2-172. BSC Protocol Data-Link Control Characters (DLCC)

Character	Description
ITB	End-of-Intermediate-Transmission Block. Separates records within a block and/or delimits field boundaries for error checking. ITB does not reverse the direction of the transmission.
NAK	Negative Acknowledgment. Sent by the receiving station to indicate that it is not ready to receive, or to request that an erroneous block be transmitted again.
RVI	Reverse Interrupt. A positive response used instead of ACK0 or ACK1; signals that the receiver must interrupt the transmission to send the transmitting station a high-priority message. The transmitting station treats an RVI as a positive acknowledgment and, in response, transmits all the data that prevents it from becoming a receiving station. The transmitting station can perform more than one block transmission to empty all its buffers. On a multipoint line, a control station can send an RVI after it receives data from a tributary station, to indicate that it wants to communicate with a different tributary station.
SOH	Start-of-Header. Signals the start of header information (ancillary information within a block).
STX	Start of Text. Signals the beginning of the text (and terminates the header information).

(Continued)

Table 2-172. BSC Protocol Data-Link Control Characters (DLCC)

Character	Description
SYN	Synchronization Character. Establishes and maintains character synchronization; also serves as filler in the absence of data or control characters. Each transmission must begin with at least two contiguous SYN characters.
TTD	Temporary Text Delay. A two-character sequence that consists of STX ENQ, which the transmitting station sends to retain the line without immediately sending data. The receiving station responds with a NAK. The TTD/NAK sequence can repeat, if the transmitter needs additional delays.
WACK	Wait-Before-Transmit Positive Acknowledgment. A positive acknowledgment that the receiver sends; signals that the receiver is temporarily unable to receive. (A receiver can send WACK as a response to a line bid on a point-to-point line or a selection sequence on a multipoint line, or as a response to data.) A receiving station can send more than one WACK until it is ready to receive. The transmitting station can respond with ENQ, EOT, or DLE EOT.

(Concluded)

If your line code is ASCII (you selected ?SASC as the line code), choose ?SLRC as the block check type. This block check type is illegal for EBCDIC transmissions.

The parity specification applies to transmission in ASCII code only. This parameter tells the operating system whether to check the parity of each character and whether to use odd parity (?SOPR) or even parity (?SEPR). If you set parity to ?SNPR, there is no parity checking.

?SEBL Continued

Connect Time-out

The connect time-out specification, offset ?STOC, defines how many seconds the operating system will wait for the line to connect before it takes the ?SEBL error return. When you enable a BSC line, the operating system first examines the state of the data set flag, DSR, which the hardware raises as soon as the line is connected. If the DSR flag is not up, the operating system suspends the ?SEBL calling task until the DSR flag is raised. If the flag is not raised within the time-out interval, the operating system takes the ?SEBL error return and passes error code ERFCT to AC0.

Sample Packet

```
PKT:    .BLK    ?SELN           ;Allocate enough space for the
                                ;packet.  Packet length = ?SELN.
        .LOC    PKT+?SCHN       ;Channel number.
        .WORD   0               ;The OS returns this value.
        .LOC    PKT+?SSTI       ;Input status word.
        .WORD   ?SDPP!?SDPR!?SASC!?SLRC!?SEPR!?SSBSC   ;Point-to-point
                                ;line type and mode, ASCII
                                ;format, even parity,
                                ;and normal BSC protocol.
        .LOC    PKT+?STOC       ;Connect time-out.
        .WORD   -1              ;30-second connect time-out (default
                                ;is -1).
        .LOC    PKT+?SMDI       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?SELN       ;End of packet.
```

Notes

- See the description of ?SDBL in this chapter to disable a BSC line.

Licensed Material - Property of Data General Corporation

?SECHR Sets extended characteristics of a device.

?SECHR
error return
normal return

Input

AC0 contains either

- Byte pointer to the name of the target device
- Channel (number) on which the device is open

AC1 contains the following flags:

bit 0 = 0	If AC0 contains a byte pointer
bit 0 = 1	If AC0 contains a channel number
bit 1 = 0	To get the current characteristics
bit 1 = 1	To get the default characteristics
bits 28–31	Length of the packet in words, between values ?CLMIN and ?CLMAX inclusive

AC2 contains the address of a ?CLMIN to ?CLMAX word area to receive the device characteristics.

Output

AC0 Unchanged

AC1 Number of characteristics words set

AC2 Unchanged

Error Codes in AC0

ERICN	Illegal channel
ERIFT	Illegal file type
ERPRE	Invalid system call parameter
ERARG	Too few or too many arguments to PMGR. You attempted to define a packet with length less than ?CLMIN.

Why Use It?

The ?SECHR call allows the owner process to change the character device's current characteristics for the duration of the calling process. It enables the calling process to select certain characteristics dynamically to tailor the target devices I/O.

Note that only the operator process (PID 2) can use ?SECHR to create new default characteristics for a character device. In addition, no process may use ?SECHR to change the split baud enable/disable characteristic: (?CSBEN/?CSBDS) or the two modem-related characteristics: modem control (?CMOD) and monitor ring indicator (?CMRI). These characteristics may only be set in the system generation dialog. Also there are characteristics fields which have no meaning on certain hardware configurations, and an attempt to set these fields will have no effect.

What It Does

The ?SECHR call may be used to set the current or default characteristics for a character device depending on the value placed in bit 1 of AC1. Since the length of the characteristic word packet is passed by the user in AC1, this call may be used to set any number of characteristics words up to ?CLMAX. ?SECHR actually serves two purposes: it allows a process to change specific characteristics words for a character device to which the caller has access, and it allows the operator process, PID 2, to set new default characteristics overriding the defaults chosen by the system generation dialog. Only the operator process (PID 2) can define new default characterisitcs.

Before issuing ?SECHR, load AC0 with the channel number of the target device or with a byte pointer to the device name. Load AC1 with the number of characteristics words to be changed. Load AC2 with the address of the "new" device characteristics words, supplied elsewhere in your program. Table 2-170 contains a complete list of character device characteristics words.

If AC0 contains a byte pointer, set bit 0 of AC1 to zero; if AC0 contains the device's channel number, set bit 0 of AC1 to 1. Bit 1 of AC1 defines whether this ?SECHR call will change the current device characteristics or the default characteristics.

You use the ?TRPE characteristic to indicate that pointer events (e.g., mouse movements) should delimit/terminate subsequent ?READs. If you are interested in retrieving pointer events, then set this characteristic.

Licensed Material - Property of Data General Corporation

?SEND — Sends a message to a console.

?SEND
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	Some
Error Codes	Some
Parameter Packet	None

Input

AC0 One of the following:

- PID of the process associated with the console
- Byte pointer to the name of the process associated with the console
- Byte pointer to the name of the console

AC1 Byte pointer to the message

AC2 Contains the following:

- For AOS/DVS only, Bit 21 is the "urgent" bit.
- Bits 22 and 23 are flag bits with the following binary values:

  00 if AC0 contains a PID

  01 if AC0 contains a byte pointer to the process name

  10 if AC0 contains a byte pointer to the console name

  11 (undefined)

- Bits 24 through 31 contain the byte length of the message (to a maximum of 256 bytes)

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Licensed Material - Property of Data General Corporation

Error Codes in AC0

ERIDT	Illegal device name type (The target device is not a console.)
ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action (AOS/DVS only; the caller tried to set Bit 21 without privilege ?PVSM)
ERVBP	Invalid byte pointer passed as a system call argument
ERMRD	Message receive disabled
ERMEM	Insufficient memory

Why Use It?

?SEND allows a process to send a message to an operator at a console without opening and closing the console. This means that you can use ?SEND to pass messages from real-time processes without consoles to system processes, such as OP CLI. Also, ?SEND allows a process to send a message to another process's console. (Normally, file I/O does not allow you to do this.)

What It Does

?SEND transmits a message from a process to a console. The terminal cannot have the ?CNRM characteristic (message receive disabled) and it cannot be waiting for a CTRL-Q because of page mode or a CTRL-S.

Before you issue ?SEND, load AC1 with a byte pointer to the message text, which may be up to 256 bytes. The operating system sends the message in the following format:

<NEW LINE>From PID xxx: <message><NEW LINE>

where:

- *PID xxx* is the PID of the process issuing the ?SEND.
- *<message>* is the message string.

You must specify the receiving console in AC0 before you issue ?SEND. To do this, supply one of the following in AC0:

- The PID of the process associated with the receiving console.
- The process name of the process associated with the receiving console (username:process_name).
- The simple pathname of the receiving device.

How you specify the receiving console determines your input to AC2.

For AOS/DVS system calls, Bit 21 is the "urgent" bit. If you set it, the operating system considers the message urgent. It ignores the setting of characteristic ?CNRM on the target console and sends the message. You must have System Manager privilege (?PVSM) to set Bit 21. If you don't set Bit 21, the operating system respects the setting of characteristic ?CNRM on the target console.

?SERMSG Returns text for associated error code. (16–Bit Processes Only)

?SERMSG
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Address of the ?SERMSG packet

Output

AC0	Length of the message in bytes
AC1	Channel number
AC2	Unchanged

Error Codes in AC0

ERICM	32–bit program incorrectly attempted to call ?SERMSG
EVTXT	Caller's input message buffer is too small
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?SERMSG allows a 16–bit program to refer to 32–bit error messages.

What It Does

?SERMSG permits 16–bit programs to return the text string associated with a 32–bit error code. Sixteen–bit programs can use ?ERMSG to return the text string associated with a 16–bit error code.

To use ?SERMSG, you must supply a packet. Figure 2–198 shows the structure and Table 2–173 lists the contents of the ?SERMSG packet.

Upon return from ?SERMSG, AC0 and AC1 contain the same return information as from ?ERMSG.

?SERMSG Continued

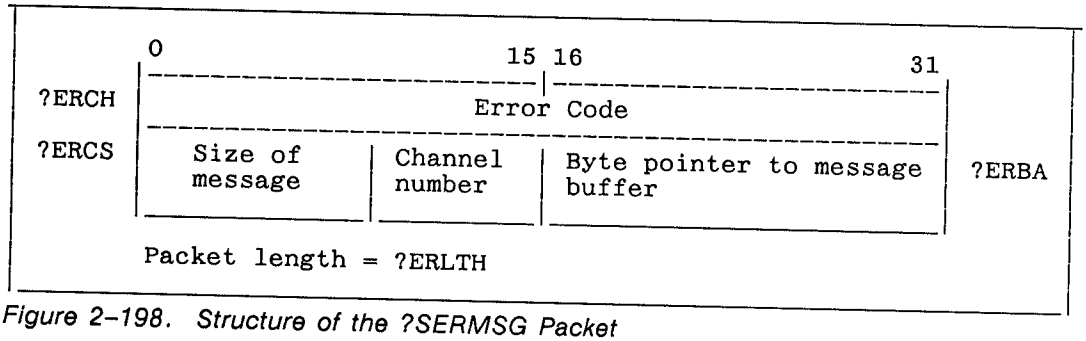


Figure 2–198. Structure of the ?SERMSG Packet

Table 2–173. Contents of the ?SERMSG Packet*

Offset	Contents
?ERCH (double word)	Error code.
?ERCS	Left byte: Size of message buffer. Right byte: Channel number. (A 377 specifies system default.)
?ERBA	Byte pointer to message buffer.

* There is no default unless otherwise specified.

Notes

- See the description of ?ERMSG in this chapter.

Licensed Material – Property of Data General Corporation

?SERT — Re-enables a relative console.

?SERT
error return
normal return

See the description of ?SDRT in this chapter for more information on ?SERT.

?SERVE

Becomes a server.

?SERVE
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Address of the ?SERMSG packet

Output

AC0	Length of the message in bytes
AC1	Channel number
AC2	Unchanged

Error Codes in AC0

No error codes are currently defined.

Why Use It?

?SERVE and ?CON are the fundamental connection-management system calls. Potential servers must issue ?SERVE to declare themselves as servers. Similarly, potential customers must issue ?CON to establish connections with existing server processes.

What It Does

?SERVE designates the calling process as a server, which can perform functions on behalf of customer processes. (Potential customers connect with the server by issuing ?CON system calls.)

Notes

- See the description of the ?CON system call in this chapter.

Licensed Material - Property of Data General Corporation

?SGES — Gets BSC error statistics.

?SGES
error return
normal return

Input

AC0	For a cumulative error tally, −1; otherwise, ignored
AC1	Channel number assigned to the line
AC2	Address of the ?SGES packet

Output

AC0	Undefined
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERLNA	I/O request for disabled line
ERNSL	Attempt to enable nonsynchronous line

Why Use It?

?SGES helps you to pinpoint problem areas on each enabled BSC line.

What It Does

?SGES returns the error statistics that the operating system records for each enabled bisynchronous (BSC) line. These statistics include the number of block check errors, the number of time-outs, and the total number of negative acknowledgment (NAK) characters received in response to send operations. If you set AC0 to −1 on input, the operating system returns a cumulative error tally. A cumulative error tally is a record of all errors that have occurred since the line was enabled or since the last ?SGES in which you did not select the cumulative error option.

Before you issue ?SGES, set up a packet of ?SGLN words, and load the packet address into AC2. Figure 2-199 shows the structure of the ?SGES packet. Table 2-174 describes each offset.

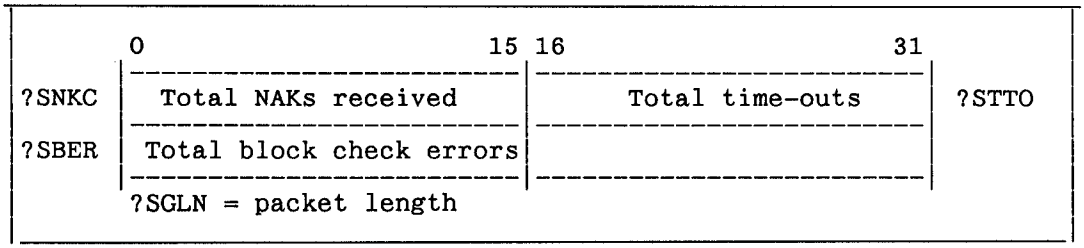


Figure 2-199. Structure of ?SGES Packet

?SGES Continued

Table 2-174. Contents of ?SGES Packet

Offset	Contents
?SNKC	Total number of NAKs received in response to a send operation. A large number of NAKs can indicate that the data was not received or, if it was received, that the OS found a block check error. (See the description of the ?SSND system call in this chapter.)
?SSTO	Total number of time-outs during receive operations. If there are no time-outs, the OS returns 0 to this offset. (See the description of the ?SRCV system call in this chapter.)
?SBER	Total number of block check errors during receive operations. These are not errors that lead to error returns from ?SRCV; rather, they are errors in the data. When an ?SRCV error occurs, the OS enters an error-recovery procedure in which it tries again to receive the data, repeatedly, if necessary. The OS may receive the data correctly on a retry.

* There is no default unless otherwise specified.

Sample Packet

```
PKT:    .BLK    ?SGLN           ;Allocate enough space for the
                                ;packet.  Packet length = ?SGLN.
        .LOC    PKT+?SNKC
        .WORD   0               ;Total number of NAKs received in
                                ;response to a send.  The OS
                                ;returns this value.  (A large number
                                ;of NAKs means that either the send
                                ;was not received or there is a block
                                ;check error.)
        .LOC    PKT+?STTO
        .WORD   0               ;Total number of time-outs during
                                ;receive.  The OS returns this value.
        .LOC    PKT+?SBER
        .WORD   0               ;Total number of block check errors
                                ;that occurred during the receive.
                                ;(These are errors in the data.  See
                                ;Table 2-174.)  The OS returns this
                                ;value.
        .LOC    PKT+?SGLN       ;End of packet.
```

Notes

- See the description of ?SCRV in this chapter for more information on receive operations.

?SIGNL

Signals another task.

?SIGNL
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	PID of the target process (The caller can specify its own PID; however, the OS will validate whatever PID is specified.)
AC2	Unique TID of the target task in the range from 1 through 32. Issue ?UIDSTAT to obtain the unique TID.

Output

AC0	Unchanged (or error code)
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPRH	Specified PID not in process hierarchy
ERSGO	Signal already outstanding
	Task does not exist

Why Use It?

You can use ?SIGNL, along with ?WTSIG and ?SIGWT, to allow tasks to synchronize access to global memory that identical local servers share.

?SIGNL Continued

What It Does

?SIGNL is similar to the IPC system call ?ISEND. However, unlike ?ISEND, ?SIGNL does not cause any data movement. Instead, ?SIGNL allows a task to signal one of the following:

- Another task in the same process.
- Another task in a different process.

A task can issue ?SIGNL, even though the target task is not waiting to receive a signal. In this case, the operating system remembers the target. Then, when the target issues a ?WTSIG system call, it does not wait. Also, a task can issue ?SIGNL from interrupt level.

Warnings

The fast interprocess signalling mechanism is used by Data General supplied software, including (but not limited to) the PMGR interface routines found in the AGENT in ring three of all processes in the operating system. And, the system makes absolutely no checks as to the legitimacy of a ?SIGNL issued by anyone or to anyone.

These two warnings have the following implications.

If a task ?SIGNLs another task, it must be absolutely sure that the next ?SIGWT or ?WTSIG issued by the receiving task will be the one it expects to receive the ?SIGNL. If such is not the case, the ?SIGNL may go to a ?SIGWT or ?WTSIG issued by another routine used by the task elsewhere in the process (e.g., the PMGR interface in ring 3, or the INFOS local server in ring 4). There is no addressing of the message as is provided by the IPC message port number or the ?REC/?XMT mailbox location.

Second, after any task resumes execution after receipt of a ?SIGNL by ?WTSIG or ?SIGWT, it is incumbent upon the receiver to verify that the condition being waited for has in fact occurred. The operating system does not authenticate ?SIGNLs sent.

Notes

- See the descriptions of ?SIGWT and ?WTSIG in this chapter.

?SIGWT

Signals another task and then waits for a signal.

?SIGWT
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	PID of the target process (The caller can specify its own PID; however, the OS validates whatever PID is specified.)
AC2	Unique TID of the target task in the range from 1 through 32. Issue ?UIDSTAT to obtain the unique TID.

Output

AC0	Unchanged (or error code)
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPRH	Specified PID not in process hierarchy
ERSCI	System call at interrupt level
ERSGO	Signal already outstanding

Why Use It?

You can use ?SIGWT, along with ?WTSIG and ?SIGNL, to allow tasks to synchronize access to global memory that identical local servers share.

?SIGWT Continued

What It Does

?SIGWT is similar to the IPC system call ?IS.R. However, unlike ?IS.R, ?SIGWT does not cause data movement. Instead, ?SIGWT allows a task to signal and then wait for a signal from one of the following:

- Another task in the same process.
- Another task in a different process.

A task can issue ?SIGWT, even though the target task or process is not waiting to receive a signal. In this case, the operating system remembers the target. Then, when the target issues a ?WTSIG system call, it does not pend. ?SIGWT will wait for a signal from another task (if one has not already been issued) before it wakes up.

Warnings

The fast interprocess signalling mechanism is used by Data General supplied software, including (but not limited to) the PMGR interface routines found in the AGENT in ring three of all processes in the operating system. And, the system makes absolutely no checks as to the legitimacy of a ?SIGNL issued by anyone or to anyone.

These two warnings have the following implications.

If a task ?SIGNLs another task, it must be absolutely sure that the next ?SIGWT or ?WTSIG issued by the receiving task will be the one it expects to receive the ?SIGNL. If such is not the case, the ?SIGNL may go to a ?SIGWT or ?WTSIG issued by another routine used by the task elsewhere in the process (e.g., the PMGR interface in ring 3, or the INFOS local server in ring 4). There is no addressing of the message as is provided by the IPC message port number or the ?REC/?XMT mailbox location.

Second, after any task resumes execution after receipt of a ?SIGNL by ?WTSIG or ?SIGWT, it is incumbent upon the receiver to verify that the condition being waited for has in fact occurred. The operating system does not authenticate ?SIGNLs sent.

Notes

- See the descriptions of ?SIGNL and ?WTSIG in this chapter.

?SINFO Gets selected information about the current operating system.

?SINFO
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Word pointer to the ?SINFO packet

Output

AC0	Undefined
AC1	Undefined
AC2	Undefined

Error Codes in AC0

ERMPR	System call parameter address error
ERPRE	Invalid system call parameter
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?SINFO is a way to find out your system's revision number, memory configuration, and current executing operating system name. If an extended call is made to return the operating system name, and the executing system is installed on the master LDU, then ?SINFO will return INSTALLED SYSTEM as the name.

?SINFO Continued

What It Does

?SINFO passes information about the version of the operating system that you are running to the packet that you specify in AC2. Figure 2-200 shows the structure of the ?SINFO packet.

Note that the packet specifications you supply are the byte pointers to two buffer areas: one area to receive the name of the master logical disk unit (LDU), and another area to receive the system identifier. Also, if the pathname to the currently executing operating system is desired, the packet offset ?SIRS must be set equal to ?PKR1, and the byte pointer to the buffer to hold the name must be included. You must reserve an area in your logical address space ?SIIL words long for the ?SINFO packet if the pathname of the executing operating system is not wanted, or else ?SIEX words long if the name is wanted.

Figure 2-200 shows the structure of the ?SINFO packet.

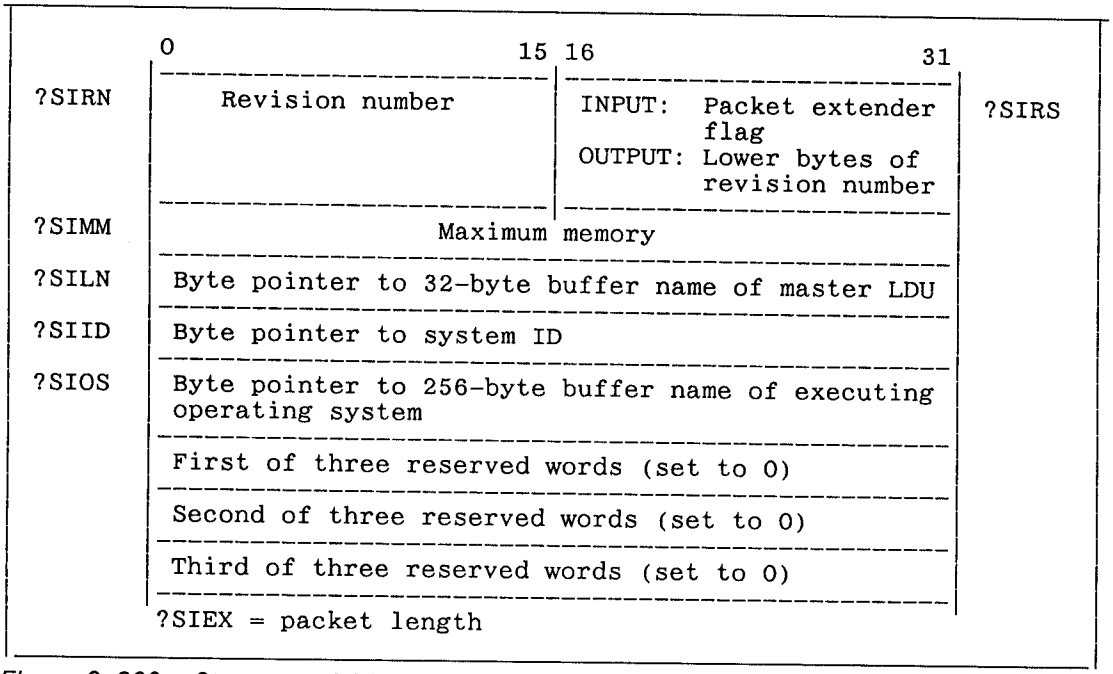


Figure 2-200. Structure of ?SINFO Packet

The operating system returns its major and minor revision number as octal values to the left and right bytes of offset ?SIRN respectively. If the revision number is 00.00.01.01, for example, ?SIRN contains 00101. The operating system returns its major and minor revision numbers as octal values to the left and right byte of offset ?SIRN respectively. If the revision number is 00.00.01.01, for example, ?SIRN contains 00101. Offset ?SIMM contains the number of the highest memory page. In a 512-Kbyte system, for example, the value of ?SIMM is octal 377; that is decimal 255. (The first page is page zero.)

If you want the operating system to return the name of the master LDU you are using and the system identifier, set up two 32-byte receiving areas in your logical address space. Use offset ?SILN as a byte pointer to the buffer area for the master LDU name, and use offset ?SIID as a byte pointer to the buffer for the system identifier, set these offsets to 0 and omit the buffer area.

If you want the operating system to return the pathname of the executing operating system, then make an extended ?SINFO call. The ?SIRS word in the packet is used to indicate an extended call on INPUT, and will contain the second 2 bytes of the system Revision Number on OUTPUT. If this word is set to ?PKR1 a new double word byte pointer has been appended to the packet. This byte pointer is to the ?MXPL (256.) byte buffer which will hold the pathname. If this word is zero, an unextended ?SINFO call is being made, and the packet does not contain the pathname byte pointer.

If either the word pointer in AC2 or the byte pointers in the packet refer to locations that are outside your logical address space, ?SINFO fails and the system returns the error code ERMPR in AC0. If ?SIRS is not set to either 0 or ?PKR1, an "Invalid system call parameter" error will result. If an extended call is made, but the byte pointer is invalid, an "Invalid byte pointer" error will result.

Sample Packet

```
PKT:    .BLK    ?SIIL           ;Allocate enough space for
                                ;the packet. Packet length
                                ;= ?SIIL.
        .LOC    PKT+?SIRN       ;Revision number.
        .DWORD  0               ;The OS returns this value.
        .LOC    PKT+?SIMM       ;Maximum memory.
        .DWORD  0               ;The OS returns this value.
        .LOC    PKT+SILN        ;Byte pointer to buffer
                                ;that contains name of
                                ;master LD.
        .DWORD  MLD*2           ;Buffer is MLD.
        .LOC    PKT+?SIID       ;Byte pointer to system
                                ;ID buffer.
        .DWORD  SID*2           ;Buffer is SID.
        .LOC    PKT+?SIOS       ;Byte pointer to buffer
                                ;that contains name of
                                ;executing op sys.
        .DWORD  SOS*2           ;Buffer is SOS.
        .LOC    PKT+?SIEX       ;End of packet.
MLD:    .BLK    16.
SID:    .BLK    16.
SOS:    .BLK    128.
```

?SLIST Sets the search list for the calling process.

?SLIST
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Byte pointer to the new search list

Output

AC0	Undefined
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERFDE	File does not exist
ERSRE	Search list resolution error (There are too many directories in the search list.)

Why Use It?

You can access directories or files outside your own file structure by setting a search list.

What It Does

?SLIST sets the search list for the calling process. The symbol ?MXPL represents the maximum length for a search list. A search list cannot contain more than eight pathnames. (Input pathnames that resolve to the same directory count as one complete pathname.)

Use the following format for the search list string:

pathname<terminator>[pathname<terminator>...]<null>

where

- terminator is either a null <000> or a NEW LINE.

You must conclude the string with a final null, e.g.,

<012>.:UDD:ANNA:SPECS<012>:UTIL<000>^MACROS<012><000>

The operating system immediately resolves each pathname, that is, replaces it with a complete pathname from the root. The final search list, after resolution, can contain more than ?MXPL bytes. (This means that you may not be able to acquire an entire pathname with the ?GLIST system call.)

Note that any pathname prefixes you use in the string are relative to the current working directory.

Notes

- See the description of ?GLIST in this chapter.

?SONS Gets a list of son processes for a target PID.

?SONS *[packet address]*
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the name of the target process
- PID of the target process
- −1 for the current process

AC1 One of the following:

- −1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID
- ignored if AC0 contains −1

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

Output

AC0 Unchanged

AC1 Unchanged

AC2 Address of the ?SONS packet

Error Codes in AC0

ERPRE	Invalid system call parameter
ERPRH	PID in AC0 points to a nonexistent process
ERPVS	Packet version not supported
ERRVN	Reserved value not zero
ERVBP	Invalid byte pointer passed as system call argument
ERVWP	Invalid address passed as system call argument

Why Use It?

Use this system call to obtain the PIDs of all son processes of a specified process.

?SONS Continued

What It Does

?SONS returns all PIDs (or at least as many as will fit in the buffer provided) that are sons of a specified process. Thus, you have a determination of a process's hierarchy because you know its subordinate processes. The returned PIDs come from the entire range of PIDS on the system.

Figure 2–201 shows the structure of ?SONS's parameter packet, and Table 2–175 describes its contents.

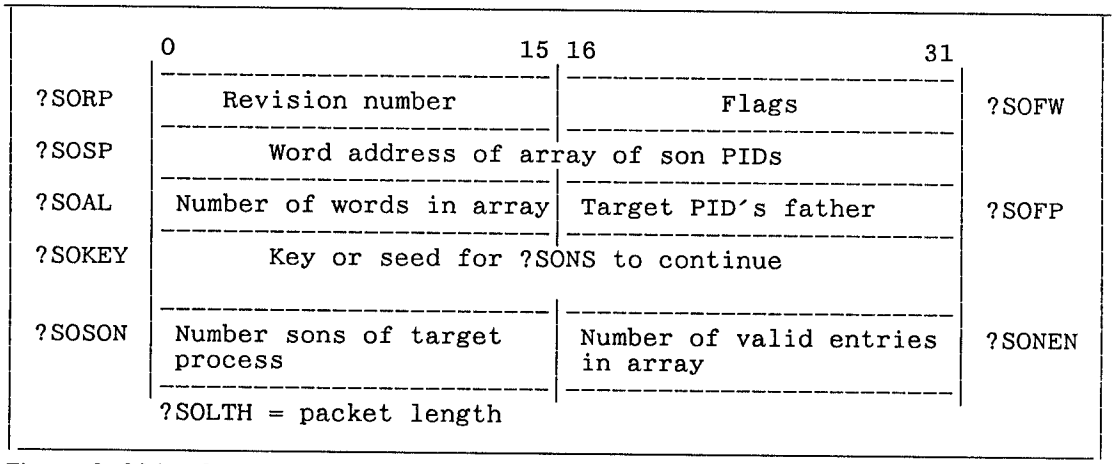


Figure 2–201. Structure of ?SONS Packet

Licensed Material – Property of Data General Corporation

Table 2-175. Contents of ?SONS Packet

Offset	Contents
?SORP	Packet revision number. Place zero here.
?SOFW	Flags. Set all unused bits to zero. Since no bit offsets are currently defined, place zero here.
?SOSP (double word)	Word address of an array in which the OS will return the list of the PIDs of the sons of the target process. Each entry in the array is one word wide.
?SOAL	Length in words of the array that receives the list of son PIDs. ?SONS will not return more PIDs than the array will hold. You must place a nonnegative number here.
?SOFP	The operating system returns the PID of the process that is the father of the target process.
?SOKEY (double word)	Set to 0 the first time you issue this call. If you cannot allocate a large enough array to contain all the PIDs of the target process, you must reissue the call to obtain the remaining PIDs. In this case, the prior issue of ?SONS has placed a number different from 0 in this offset; leave it here for all reissues. Don't keep the returned key number indefinitely, because it only is a snapshot of the current process hierarchy. Since this hierarchy typically changes frequently, the key probably isn't valid after significant processing time has elapsed.
?SOSON	The OS returns the number of PIDs of processes that were sons of the target process when ?SONS executed. You can use this number to adjust the size of the array. The OS always returns ?SOSON, even if ?SOAL contains 0.
?SONEN	The OS returns the number of valid and nonzero PIDs that it stored in the array. A program that uses this number doesn't have to count the number of nonzero entries to determine how many son PIDs were returned.

?SOPEN

Opens a file for shared access.

?SOPEN
error return
normal return

Input

AC0	Byte pointer to the target file's pathname
AC1	One of the following: • Target file's channel number • −1 to allow the OS to assign a channel number. (This is the default.)
AC2	One of the following: • 0 to open this file for read-only purposes • Any other value to open the file for read/write purposes

Output

AC0	Unchanged
AC1	Target file's channel number
AC2	Unchanged

Error Codes in AC0

ERILO	Illegal open (You tried to ?SOPEN a file whose element size is not 4 or a multiple of 4.)
ERVBP	Invalid byte pointer passed as a system call argument

File system error codes

Licensed Material - Property of Data General Corporation

Why Use It?

?SOPEN opens a file for shared access. In general, you would use ?SOPEN and the other shared-page system calls to enable several processes to share the same re-entrant code or the same data. This reduces disk I/O and conserves memory.

What It Does

?SOPEN opens a file for shared access. For the shared-page mechanism to work correctly, the file you open must have an element size of 4 or a multiple of 4.

To restrict the caller's access privileges to the file, load AC2 with 0. This allows the calling process read-only access, which prevents the caller from modifying the file, regardless of the file's ACL.

If AC2 specifies both read and write, the operating system flushes the modified contents of the file to disk as soon as the caller issues ?SCLOSE to close the file.

Notes

- See the descriptions of ?SCLOSE, ?OPEN, and ?GCLOSE in this chapter.

?SOPPF

Opens a protected shared file.

```
?SOPPF
error return
normal return
```

Input

AC0	One of the following: • Byte pointer to the filename on first open • 0 on subsequent opens
AC1	One of the following: • Channel number • -1, to indicate that you want the OS to define the channel number
AC2	Packet address

Output

AC0	Unchanged
AC1	Channel number
AC2	Unchanged

Error Codes in AC0

EREO1	File is open, can't exclusively open
EFFAD	File access denied

Why Use It?

?SOPPF is the system call that opens a protected shared file. When the ?PFFO flag is set in offset ?PFFLG, ?SOPPF allows a segment image to open a shared file in such a way that no other segment can open the file without its explicit permission. When the ?PFFO flag is not set in offset ?PFFLG, ?SOPPF allows a segment image to open a protected shared file using the access privileges granted by the first opener of the file.

What It Does

?SOPPF allows you to open a shared file in a protected manner. Then, the caller can issue shared-page system calls, just as if the channel had been opened with ?SOPEN. To close a shared file that has been opened with ?SOPPF, issue ?SCLOSE, just as if it had been opened with ?SOPEN.

To use ?SOPPF, you must supply a packet. Figure 2-202 shows the structure and Table 2-176 describes the contents of the ?SOPPF packet.

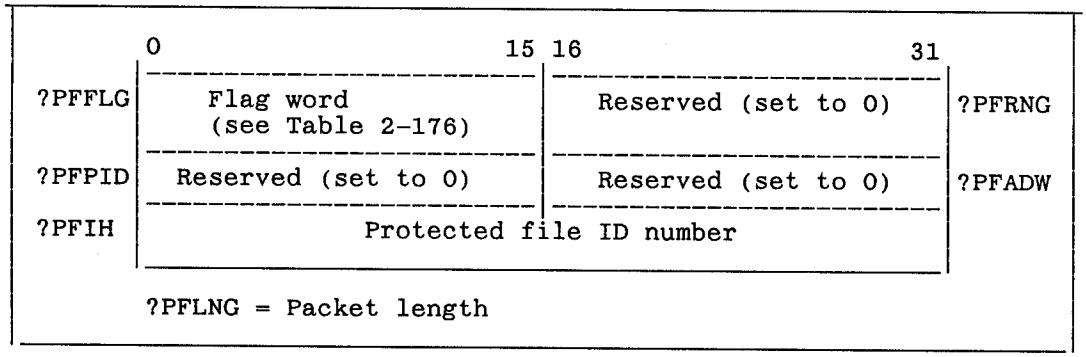


Figure 2-202. Structure of ?SOPPF Packet

Table 2-176. Contents of ?SOPPF Packet*

Offset	Meaning
?PFFLG	Flag word. ?PFFO--First open. ?PFRW--Open for read/write.
?PFRNG	Reserved. (Set to 0.)
?PFPID	Reserved. (Set to 0.)
?PFADW	Reserved. (Set to 0.)
?PFIH (double word)	Protected file ID number.

* There is no default unless otherwise specified.

If you set the ?PFFO flag in offset ?PFFLG, then the caller must be the first to open the file. In this case, ?SOPPF causes the following to occur:

- The caller (user) passes a byte pointer to a pathname as input.
- If the file is already open, the operating system returns the error code EREO1.
- The operating system returns an ID number that uniquely identifies the protected shared file.
- The operating system returns a channel number for use by the caller.
- The operating system grants the caller whatever access privileges are indicated by the shared file's ACL.
- The operating system prevents the shared file from being opened, except by another ?SOPPF.

?SOPPF Continued

This first opener uses the ?PMTPF system call to permit other segment images to open the file. Only the first opener of a protected shared file can issue a ?PMTPF against the shared file. (See the description of ?PMTPF in this chapter for more information about access privileges.)

If the ?PFFO flag in offset ?PFFLG is not set, ?SOPPF causes the following to occur:

- The caller (user) passes the unique protected file ID as input.
- The operating system returns an error if the caller does not have explicit permission from the first opener of the shared file to open it.
- The operating system returns a channel number for the caller to use.
- The operating system grants the caller those access privileges that the first opener's ?PMTPF specified.

Notes

- See the description of ?PMTPF in this chapter.

?SPAGE — Performs a shared–page read.

?SPAGE *[packet address]*
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Unchanged
AC2	Address of the ?SPAGE packet

Error Codes in AC0

ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can use ?SPAGE to move a page or pages into the shared area of your logical address space. Offset ?PSTI in the ?SPAGE packet allows you to restrict the calling process to read access only. This is a way of preventing modifications to shared routines or shared data.

What It Does

?SPAGE validates a shared page or pages in the caller's logical address space. Note that you must open the target file with ?SOPEN before you can issue ?SPAGEs against it.

If the target page is already on the LRU chain and, therefore, memory resident, the operating system checks the caller's access to it, and adds the page to the working set when the caller references it. If the page is not in memory, the operating system reads it into memory before it adds the page to the working set (at the time the page is referenced).

Specify the number of blocks you want to read or write in the right byte of offset ?PSTI. Bit ?SPRO in the left byte of ?PSTI is used to limit access to the file to Read access only.

?SPAGE Continued

Figure 2–203 shows the structure of the ?SPAGE packet, and Table 2–177 describes its contents. Note that the packet is identical to the packet for the block I/O calls ?RDB and ?WRB, which are described in this chapter.

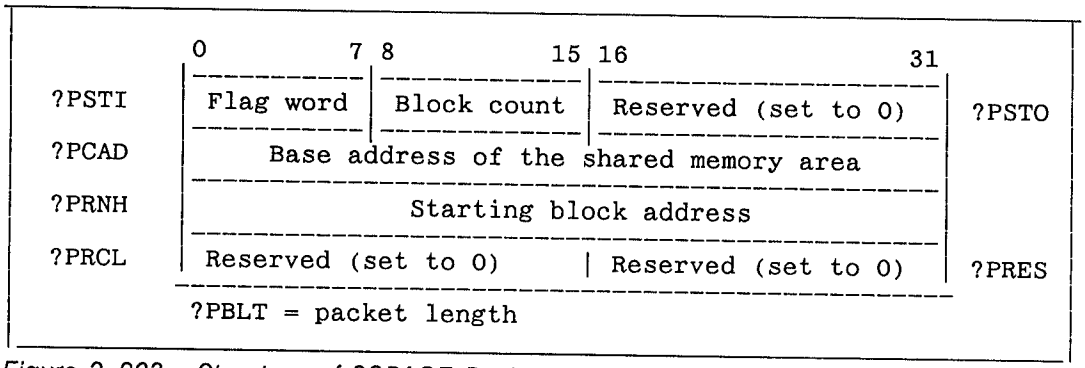


Figure 2–203. Structure of ?SPAGE Packet

Table 2–177. Contents of ?SPAGE Packet*

Offset	Contents
?PSTI	Left byte contains the following flag: ?SPRO -- allow Read access only. Set to 1 if you want to write protect the page(s). (This will even write protect pages opened with ?SOPEN.) Right byte contains the block count. The block count is the number of disk blocks to be transferred into the shared area. This number must be 4 or a multiple of 4.
?PSTO	Reserved. (Set to 0.)
?PCAD (double word)	Starting address in memory; that is, the base address of the shared memory area to use. (This is always the start of a page boundary.)
?PRNH (double word)	Address of the file's starting block.
?PRCL	Reserved. (Set to 0.)
?PRES	Reserved. (Set to 0.)

* There are no defaults unless otherwise specified.

Licensed Material – Property of Data General Corporation

Sample Packet

```
PKT:    .BLK    ?PBLT           ;Allocate enough space for the
                                ;packet.  Packet length = ?PBLT.
        .LOC    PKT+?PSTI       ;Number of disk blocks to read into
        .WORD   4               ;shared area (must be 4 or a multiple
                                ;of 4).  Transfer four blocks.
        .LOC    PKT+?PSTO       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?PCAD       ;Base address of shared memory area.
        .DWORD  BUFFER          ;BUFFER is address of data buffer.
        .LOC    PKT+?PRNH       ;Starting block address.
        .DWORD  0               ;Block 0 or first block of file.
        .LOC    PKT+?PRCL       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?PRES       ;Reserved.
        .WORD   0               ;You must set this value to 0.
        .LOC    PKT+?PBLT       ;End of packet.
```

Notes

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

?SPOS

Sets the position of the file pointer.

?SPOS
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the ?READ or ?WRITE packet

Error Codes in AC0

ERBOF	Positioning before beginning of the file
EREOF	End of file
ERFNO	Channel not open
ERICN	Illegal channel
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can use ?SPOS to change the file pointer's position in a file without doing I/O.

What It Does

?SPOS repositions the file pointer of a file that is currently open for I/O (an open file or a file you are currently reading or writing). The operating system maintains a separate file pointer for each open channel to keep track of I/O performed across the channel.

By default, the file pointer moves forward sequentially as the operating system executes each read or write. You can position the file pointer before an I/O sequence by manipulating the file pointer specifications in the I/O packet. Also, you can issue ?SPOS to change the position of the file pointer without performing I/O. The file pointer parameters are

- ?IRNH, which is a double-precision integer that indicates the position of the file pointer.
- ?IRCL, which states the maximum number of bytes to read or write.
- Bit ?IPST in offset ?ISTI, which determines whether the value in ?IRNH is an absolute or a relative value.

Licensed Material - Property of Data General Corporation

You manipulate these parameters with ?SPOS just as you would with ?READ or ?WRITE. Table 2-178 summarizes the file-pointer settings. Note that you cannot issue ?SPOS against a labeled magnetic tape file.

Table 2-178. File-Pointer Settings

?IPST Value	?IRNH Value	Pointer Position
0	0	At the next record.
0	n	For fixed-length records, at relative record offset n (forward or backward from the current file pointer position); for all other record types, at relative byte offset n.
1	0	At the beginning of the file.
1	-1	At the end of the file.
1	n (any integer)	For fixed-length records, at the nth record; for all other types, at the nth byte. (The first record or byte is 0.)

If the record type is fixed-length, set offset ?IRCL to the record length or to -1, the default value. Use the default only if you specified the record length in the ?OPEN, ?READ, or ?WRITE packet. For all other record types, set ?IRCL to 0. Set bit ?IPST to 1 if you want the operating system to interpret ?IRNH as an absolute value. Set ?IPST to 0 if you want the operating system to interpret the file pointer position as a relative value.

See the description of ?READ/?WRITE in this chapter for more information on setting the file pointer's position.

Notes

- See the descriptions of ?OPEN, ?READ, and ?WRITE in this chapter.

?SRCV

Receives data or a control sequence over a BSC line.

?SRCV
error return
normal return

Input

AC0	(Used by multipoint control stations only) One of the following: • −1 for a general poll • Byte pointer to a poll address for a specific poll	
AC1	Channel number assigned to the line	
AC2	Address of the ?SRCV packet	

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERCRC	CRC check (Indicates a cyclical redundancy block check error.)
ERDIS	Disconnect occurred on a switched line
ERENQ	ENQ received after time-out (The OS detected an ENQ from the sending station after it reached the limit for retries.)
EREOT	EOT character received
EREPL	End of polling list reached
ERISE	Input status error
ERLIS	Line in session (You tried to issue two receive initial system calls in a row.)
ERLNA	Attempt to enable nonsynchronous line
ERNAK	Transmission failure (The OS received a NAK while it was waiting for a line bid, and it has reached its limit for retries (point-to-point lines only).)
ERNPL	Polling list not defined (This error occurs only on multipoint lines.)
ERNSL	Device associated with the channel number is not a synchronous line
ERSCS	You tried to issue a receive continue without a prior receive initial
ERSIM	There is an outstanding system call on this line
ERTOF	Time-out value exceeded
ERTRF	Transmitter failure
ERUNI	Unknown or inappropriate response received
ERVBP	Invalid byte pointer passed as a system call argument

Licensed Material - Property of Data General Corporation

Why Use It?

You can use ?SRCV whenever you want to receive data over an enabled BSC line.

What It Does

?SRCV prepared the calling station to receive a block of data or data-link control characters over an enabled BSC line. Before you issue ?SRCV, perform the following steps:

1. Reserve a receive buffer in your logical address space for the incoming data or control sequence.
2. Set up the ?SRCV packet in your logical address space.
3. Load AC1 with the BSC line's channel number.
4. Load AC2 with the packet address.

Figure 2-204 shows the structure of the ?SRCV packet, and Table 2-179 describes each offset.

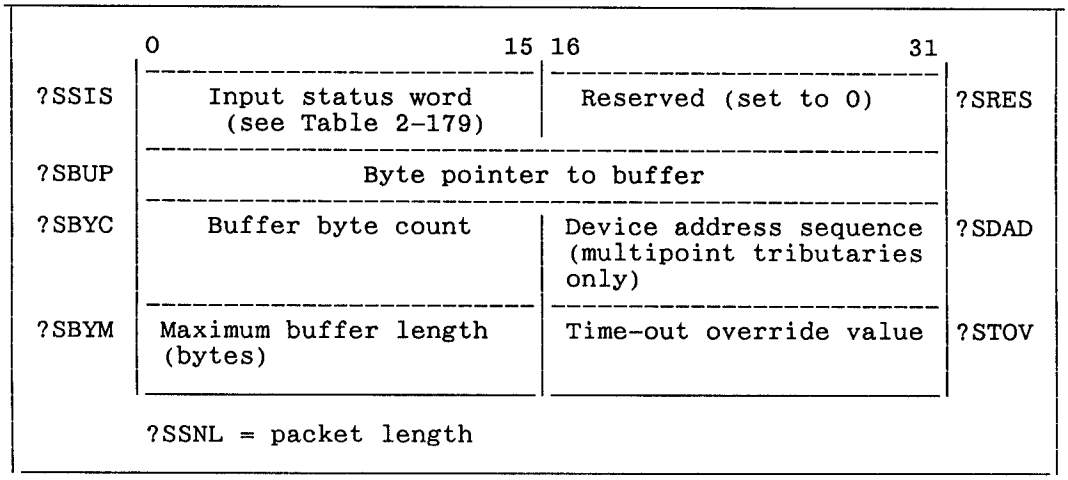


Figure 2-204. Structure of ?SRCV Packet

Table 2-179. Contents of ?SRCV Packet*

Offset	Contents	Input Value	Output Value
?SSIS	Input status word:		
	Text mode.	None.	?TRAN--If received transparent data.
	Block start.	None.	?STXB--STX received. ?SOHB--SOH received.
	Intermediate text block indicator.	None.	?SITB.
	Multipoint information.	None.	?SPLR--Your system was polled. ?SSLR--Your system was selected.
	Response type (receive continue only)		
	(Directs OS to respond to the sender with the specified control character.)	?SRVI--Reverse interrupt. ?SNAK--NAK. ?SACK--ACK0 or ACK1. ?SAK0--Override ACK1, send ACK0. ?SAK1--Override ACK0, send ACK1.	Unchanged.
	Call type.	?CINT--Receive initial. ?CONT--Receive continue.	If ?CINT was set and the bid for the line was succesful, the call type is changed to ?CONT.

* There is no default unless otherwise specified.

(Continues)

Table 2-179. Contents of ?SRCV Packet*

Offset	Contents	Input Value	Output Value
?SSIS (Cont.)	Call type. (Cont.)	The following apply to receive continues from multipoint tributaries only: ?SWAK--Send WACK; then wait for poll or select. ?SPET--Send EOT; then wait for poll or select. ?SPRV--Send RVI; then wait for poll or select. ?SPNK--Send NAK; then wait for poll or select.	
	Component Selection.	None.	?SDAC--Received device address characters.
?SRES	Reserved.	Set to 0.	N/A.
?SBUP (double word)	Buffer byte pointer.	Byte pointer to a receive buffer for the data.	Unchanged.
?SBYC	Buffer byte count.	None.	Number of bytes received.

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

Table 2-179. Contents of ?SRCV Packet*

Offset	Contents	Input Value	Output Value
?SDAD	Device address sequence.	None.	Left-justified device address, 2-character maximum. (You receive the device address immediately after your poll or select address.) The OS returns this value to a multipoint tributary only after an ?SRCV initial call.
?SBYM	Maximum buffer length.	Byte length of receive buffer.	Unchanged.
?STOV	Time-out override value. DEFAULT = -1 (8	Length of time the OS will wait for the remote station to respond, before it enters its BSC recovery procedure. seconds)	Unchanged.

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

There are two types of ?SRCV system calls:

- Receive initial calls, which open the transmission.
- Receive continue calls.

The operating system executes these two types of ?SRCVs differently, depending on whether the caller is a station on a point-to-point line, or the control station or a tributary on a multipoint line. The following discussion is divided to reflect these differences.

Note that you must set certain parameters regardless of the kind of station you are calling from. The next section describes these parameters.

Required Input

Set offset ?SBUP to the address of the receive buffer you set aside for the data, and set offset ?SBYM to the length of the receive buffer.

In offset ?SSIS, the input status word, specify the type of receive you want to perform. The ?CINT mask defines the ?SRCV as a receive initial call, and the ?CONT mask defines it as a receive continue. Be sure to set the correct mask. You cannot issue two receive initial calls in a row, nor can you issue a receive continue before you issue a receive initial call.

If you set ?CINT and the bid for the line is successful, the call type field is changed to ?CONT. As a result, should you get an error, you can check whether the error occurred during the bid for the line or during reception of the data block.

Receive Continue Calls

For receive continue calls, you must set one of the response-type masks in offset ?SSIS. The response-type field defines the control character the operating system uses to reply to the sending station after each transmission. The response-type options are

- Return a negative acknowledgment (?SNAK) to the sender.
- Return a positive acknowledgment (ACK0 or ACK1) in the correct alternating sequence (?SACK) to the sender.
- Return an ACK0 positive acknowledgment (?SAK0) to the sender.
- Return an ACK1 positive acknowledgment (?SAK1) to the sender.
- Return a reverse interrupt (?SRVI) to the sender.

Note that there are three variations for a positive acknowledgment. Under standard BSC protocol, the operating system alternates the ACK0 and ACK1 characters; that is, if a station uses ACK0 as its first positive acknowledgment, it will use ACK1 as its second positive acknowledgment, and vice versa. The ?SAK0 and ?SAK1 options let you specify a single positive acknowledgment character for the duration of the ?SRCV.

?SRCV Continued

Response type ?SRVI directs the operating system to reply with a reverse interrupt (RVI) character. Use this response type if you want to interrupt the transmission to pass the sending station a high-priority message.

You can continue to issue receive continue calls until you have obtained all the sending station's data. You will know the transmission has ended when the operating system returns error code EREOT in AC0.

If you issue a receive continue call and the sending station fails to respond or responds inappropriately, the operating system enters its BSC error-recovery procedures.

?SRCV from a Point-to-Point Station

If you issue ?SRCV as a receive initial call from a point-to-point station, the operating system waits for a bid sequence from the other station before it executes ?SRCV. By default, the operating system waits 8 seconds. You can set a different time-out value in offset ?STOV.

If the other station fails to send a bid sequence in time, the operating system enters the BSC error-recovery procedure for receive initial calls. If the operating system receives the other station's bid in time, the ?SRCV succeeds, and the operating system returns the first block to your receive buffer.

?SRCV from a Multipoint Control Station

When you issue ?SRCV from a control station on a multipoint line, the operating system performs either a general poll to solicit data from all your enabled tributaries or a specific poll to solicit data from one tributary. Your input to AC0, before you issue ?SRCV, determines what kind of polling the operating system performs.

On a receive-initial general poll, the operating system sends each entry in the polling list down the BSC line, starting with the entry for the first relative console (tributary). If a tributary has no data to send, it responds to its poll address with an EOT character, and the operating system continues polling.

A general polling operation ends when one of the following occurs:

- A tributary sends data to the control station. (The operating system then places the data in the control station's receive buffer and loads AC1 with the sender's relative console number.)
- The operating system reaches the end of the polling list without receiving data. (The operating system returns error code EREPL in AC0.)
- A tributary fails to respond within the time value that you specify in packet offset ?STOV. (The operating system returns error code ERTOF in AC0), or a tributary responds inappropriately. The operating system returns error code ERUNI in AC0.)

Receive continue calls from multipoint stations work the same as receive continue calls from point-to-point stations. (See the "Receive Continue Calls" section for details.)

?SRCV from a Multipoint Tributary

An ?SRCV receive initial call issued from a tributary signals the operating system that the tributary is prepared to accept data from the control station. The operating system responds by monitoring the line for your tributary's poll or select address. If the control station fails to send the poll or select address within the interval you set in ?STOV, ?SRCV fails and the operating system returns error code ERTOF in AC0.

If your tributary was polled in time, the operating system sets bit ?SPLR in offset ?SSIS and takes the normal return from the ?SRCV. You should respond by issuing an ?SSND send continue call if you have data to send, or by issuing an ?SSND EOT (end-of-transmission) if you do not have data to send.

If your tributary was selected in time, the operating system sets bit ?SSLR in offset ?SSIS and takes the normal return. If you are ready to receive data, respond to the select sequence with an ?SRCV receive continue that has one of the positive response bits set (?SACK, ?SAK0, or ?SAK1). ?SAK0 and ?SAK1 override the operating system's usual alternation between ACK0 and ACK1, and select a specific positive response (ACK0 or ACK1) for this receive call.

These parameters also reset the operating system's alternation pattern. For example, if you select ?SAK0, the next positive response will be ACK1, unless you override the alternation pattern again.

?SRCV Continued

If you are not ready to receive data, issue a ?SSND continue with the call type set to ?SWAK (wait-before-transmission, positive acknowledgment).

?SRCV Output Values

A normal return from ?SRCV causes the operating system to set certain masks in offset ?SSIS. These masks indicate the text mode of the transmission and the presence of several data-link control characters or poll, select, or device addresses. The previous section describes two of these masks, ?SPLR and ?SSLR. Table 2-180 lists all of the masks and describes their meanings.

Table 2-180. Masks Returned on ?SRCV System Calls

Mask	Information Returned	Meaning
?TRAN	Text mode.	If set, transparent text received.
?SOHB		If ?SOHB is set, start of header (SOH) character received.
?STXB	Start of block.	If ?STXB is set, start of text (STX) character received.
?ETBB		If ?ETBB is set, start of text (STX) character received.
?ETXB	End of block.	If ?ETXB is set, end of text (ETX) character received.
?SITB	ITB received.	If set, end of intermediate transmission block (ITB) character received.
?SPLR	Poll indicator.	If ?SPLR is set, poll address received.
?SSLR	Select indicator.	If ?SSLR is set, select address received.
?SDAC	Device/component selector.	If set, device address received.
?CONT	Call type.	If set, ?CINT resulted in successful bid for the line.

Licensed Material - Property of Data General Corporation

The operating system sets the ?SITB flag when one or more ITB (intermediate text block) characters delimit certain portions of the text block. When a block contains these characters, the operating system precedes each intermediate text block portion with a value that indicates its length. Figure 2–205 shows a receive buffer that consists of three intermediate text block portions with lengths of 4, 5, and 6 characters, respectively.

Word	Receive Buffer	
0	4	A
1	B	C
2	D	5
3	E	F
4	G	H
5	I	6
6	J	K
7	L	M
8	O	P

Figure 2–205. ITB Receive Buffer Format

Notes

- See the description of ?SSND in this chapter.

?SSHPT — Establishes a new shared partition.

?SSHPT
error return
normal return

Input

AC0	Logical page number of the first page in the shared memory area (range: 1 through 262,143)
AC1	Number of pages in the new shared area
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERMEM	Insufficient memory available (You may have tried to overlap the shared area into the unshared area.)
ERSHP	Error on shared partition set

Why Use It?

You might issue ?SSHPT before you issue ?SOPEN and ?SPAGE. This would expand the shared area of your logical address space before calling in one or more shared pages. A companion call, ?GSHPT, returns the current size of the shared area.

What It Does

?SSHPT either increases or decreases the caller's valid shared-page partition. Pages can only be shared within an established boundary. ?SSHPT effectively adds or removes space for potential shared pages.

As the error messages suggest, the current shared and unshared boundaries are important factors in the success of an ?SSHPT. The operating system takes the error return and passes error codes to AC0 under the following conditions:

- When you try to overlap the shared area of the logical address space with the unshared area (error code ERMEM).
- When you use ?SSHPT to transfer unshared pages to the shared area (error code ERSHP).
- When you try to release shared pages you are currently using (error code ERSHP).

Notes

- See the descriptions of ?SOPEN, ?SPAGE, and ?GSHPT in this chapter.

Licensed Material - Property of Data General Corporation

?SSID

Sets the system identifier.

?SSID
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Byte pointer to a 32-byte buffer that contains the ASCII string to use as the system identifier

Output

AC0	Undefined
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERMPR	System call parameter address error
ERPRE	Invalid system call parameter
ERPRV	Caller not privileged to make this call
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

Typically, operators set the system identifier for each computer in their installation. EXEC returns the system identifier as part of the message:

TYPE NEW-LINE TO BEGIN LOGGING ON

EXEC looks at the system identifier only once — when EXEC is created. Subsequent changes to the system identifier have no effect.

To find out what the system identifier is for your current system before you issue ?SSID, issue ?GSID.

What It Does

?SSID assigns a unique system identifier to your current system. Only the operator process (PID 2), or a process that has System Manager privilege, can issue ?SSID.

Before you issue ?SSID, load AC2 with the address of the system identifier (defined elsewhere in your program). The system copies the string you specify into the system SYSID buffer. By convention, anything appearing after the first null (if any) in the system identifier is ignored by most processes.

Notes

- See the description of ?GSID in this chapter.

?SSND

Sends data or a control sequence over a BSC line.

?SSND
error return
normal return

Input

AC0	Byte pointer to a tributary's select address and device address (used by multipoint control station, only)
AC1	Channel number assigned to the line
AC2	Address of the ?SSND packet

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Licensed Material - Property of Data General Corporation

Error Codes in AC0

ERBNK	Bid error (too many NAKs) (The ?SSND caller received a NAK in response to its line bid (point-to-point), or its select sequence (multipoint control station).)
ERCNV	Conversational reply received
ERCRC	CRC check
ERCTN	Contention situation while bidding (Both stations on a point-to-point line tried to bid at the same time.)
ERDIS	Disconnect received on a switched line (This error also occurs when a ?SSND caller receives a DLE EOT in response to its TTD sequence.)
EREOT	EOT character received (The ?SSND caller received an end of transmission character in response to its data block or WACK character.)
ERLIS	Line in session on ?SSND (You tried to issue two ?SSND initial calls in a row.)
ERLNA	I/O request for disabled line
ERNAK	Transmission failure (NAK count) (The OS received a NAK in response to a data block, or a NAK WACK sequence from the receiver; this also occurs when the sending station receives a NAK in response to an ENQ (repeat previous response), and the retry limit is exceeded.)
ERNSL	Attempt to enable nonsynchronous line
ERRVI	RVI response received (An RVI character received in response to the sender's bid (point-to-point only) or select sequence (multipoint only), or following the receipt of a WACK.)
ERSCS	?SSND continue without line in session (You tried to issue an ?SSND continue without a previous ?SSND initial.)
ERSIM	There is an outstanding call on this line
ERTOF	Transmission failure (The time-out value has elapsed.)
ERTRF	Transmitter failure
ERUNI	Uninterpretable response received
ERVBP	Invalid byte pointer passed as a system call argument
ERWAK	A WACK character received in response to a line bid (point-to-point only) or a select sequence (multipoint only)

Why Use It?

?SSND allows you to send a block of data over an enabled BSC line.

What It Does

?SSND lets the calling station send a block of data or data-link control characters over an enabled BSC line. A process can issue ?SSND to bid for the BSC line (send initial), to bid for the line and then send data (send continue), or to send data (send continue).

Before you issue ?SSND, set up a packet in your logical address space, load AC2 with its address, and load AC1 with the line's channel number. Figure 2-206 shows the structure of the ?SSND packet, and Table 2-181 describes each offset.

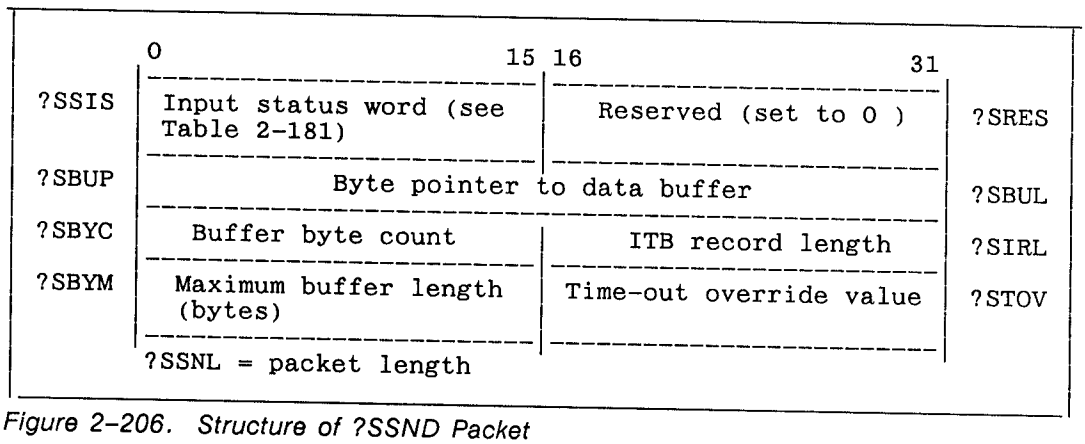


Figure 2-206. Structure of ?SSND Packet

Licensed Material - Property of Data General Corporation

Table 2-181. Contents of ?SSND Packet*

Offset	Contents
?SSIS	Input status word. Data Type. ?NTRN--Nontransparent Text Mode. ?TRAN--Transparent Text Mode. DEFAULT = ?NTRN (Nontransparent Text Mode). Block Start. ?STXB--Start of text. ?SOHB--Start of header. DEFAULT = ?STXB (start of text). Block Type. ?ETBB--End of block. ?ETXB--End of text. DEFAULT = ?ETBB (end of block). Send Intermediate Text Blocks. ?SITB. DEFAULT = No intermediate text blocks. Accept Conversational Replies. ?SCON. DEFAULT = Do not accept conversational replies. Send Device Address Characters (for select sequences only). ?SDAC. DEFAULT = Do not send device address.

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

Table 2-181. Contents of ?SSND Packet*

Offset	Contents
?SSIS (Cont.)	?CINT--send initial (with data). ?CONT--send continue (with data). ?SWAK--send WACK sequence. ?SEOT--send EOT sequence. ?SDET--send DLE ETB or DLE ETX sequence (to signal end of transparent text mode). ?STTD--send TTD sequence (to signal temporary text delay). ?SINT--send RVI. ?SDIS--perform logical disconnect.
?SRES	Reserved. (Set to 0.)
?SBUP (double word)	Buffer byte pointer. This is a byte pointer to the buffer in which the data you are sending resides.
?SBYC	Buffer byte count. This is the number of bytes you wish to send.
?SIRL	ITB record length. This is the length (in bytes) of the intermediate text blocks, which are records delimited by an ITB character. DEFAULT = 0 (no ITBs).
?SBYM	Maximum buffer length. This is the byte length of the data buffer.
?STOV	Time-out override value. This is the length of time the O S will wait for a reply before it begins the BSC error-recovery procedures. DEFAULT = -1 (8 seconds).

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

Input Status Word

Offset ?SSIS, the input status word, defines certain characteristics of the block you will send.

The data type field specifies whether the data is in Transparent Text Mode (?TRAN), or Nontransparent Text Mode (?NTRN). Under transparent text mode, most data-link control characters are treated only as bit patterns; that is, they have no control significance. The exceptions are DLE STX, which signals the beginning of Transparent Text Mode, and DLE ETB or DLE ETX, which signals the end of Transparent Text Mode.

Although the operating system inserts the necessary data-link characters, including the BCC character, when you issue ?SSND, three fields in offset ?SSIS let you specify the data-link characters you want to use.

The block start field specifies whether the operating system should preface the block with an STX (start of text) character or with an SOH (start of header) character. Set mask ?SOHB if you are sending header information in this transmission and set mask ?STXB if you are sending text. The ?STXB mask is the default.

The block type field directs the operating system to append either an ETB (end of text block) or an ETX (end of text) character to the data block. The ETX character terminates the last text block in a message. Thus, you should set mask ?ETXB if this is the last block you are sending. The default for this field is ?ETBB. This mask corresponds to the ETB character, which must terminate every text block except the last one.

The send intermediate text block field directs the operating system to delimit portions of the data block with intermediate transmission (ITB) characters. If you select this option (by setting mask ?SITB), you must set offset ?SIRL to the length of the intermediate text blocks. If you set ?SIRL to 4, for example, the operating system places an ITB character after every fourth byte in the text block. The operating system ignores the contents of ?SIRL if bit ?SITB is not set.

Select the ?SCON mask if you want to accept conversational replies from the receiving station. Conversation mode allows the receiver to respond to the block you send with header information or data, instead of a positive ACK0 or ACK1 sequence. Note that the block you send must be a complete data block (not a data-link sequence) for conversational mode to work. In general, you should not give a conversational reply to a header block or a block that ends with an ETB character.

?SSND Continued

The call type field defines ?SSND as one of the following types:

- ?CINT, which is a send initial with data. (If the bid for the line is successful, the operating system changes this to ?CONT.)
- ?CONT, which is a send continue with data.
- ?SWAK, which is a wait-before-transmission (positive acknowledgment).
- ?SEOT, which is an end-of-transmission.
- ?SDET, which is a line disconnect call.
- ?STTD, which is a temporary text delay.

Table 2-182 describes each type of call.

Other ?SSND Offsets

Offsets ?SBUP and ?SBUL point to the buffer you reserved in your logical address space for the data you will send. Note that the buffer byte count you supply in offset ?SBYC must be for the entire contents of the data buffer. If the data consists of intermediate text blocks (defined by ?SITB and ?SIRL), the data buffer should contain an exact multiple of the intermediate block length you specified in offset ?SIRL. Otherwise, the last intermediate text block will be shorter than the others, because the operating system does not pad the last intermediate text block.

If you set both ?SBUP and ?SBYC to 0 on a send initial, the operating system interprets your send call as a bid for the line. If the bid succeeds, the operating system simply takes the ?SSND normal return, without sending data. If you set either of these parameters to 0, but not both, the operating system takes the ?SSND error return and returns one of the following error codes in AC0:

Error Code	Input
ERBPE	?SBUP = 0 ?SBYC = any nonzero value
ERBCT	?SBUP = any nonzero value ?SBYC = 0

Licensed Material - Property of Data General Corporation

When you select the conversational reply option (?SCON), the operating system stores the response data from the receiver in the data buffer defined by ?SBUP, and records the length of the response data in offset ?SBYC.

Table 2-182. ?SSND Call Types

Station Type	Call Type	Action
Point-to-point	Send initial (with data) (?CINT)	Send ENQ (line bid) to the receiving station, and wait for an ACK0 response. - If line bid successful, ?CINT is changed to ?CONT; otherwise, remains the same. - If no ACK0 within time-out interval (?STOV), then begin BSC error-recovery procedures. If ACK0 within the time-out interval, then ?SSND data block, wait for ACK1 response. - If no ACK1 within time-out interval, then begin BSC error-recovery procedures. If ACK1 within time-out interval, then take ?SSND normal return.
	Send continue (with data) (?CONT)	Send data to receiving station, and await correct alternating response (ACK0 or ACK1). - If no response within time-out interval, then begin BSC error-recovery procedures.
	Send continue (with data) (?CONT)	- If correct response, then take ?SSND normal return.
All types (point-to-point and multipoint)	WACK sequence	Send WACK to receiver, and wait for an ENQ reply. - If no ENQ within time-out interval, then begin BSC error-recovery procedures. - If ENQ within time-out interval, then take ?SSND normal return.

(Continues)

?SSND Continued

Table 2-182. ?SSND Call Types

Station Type	Call Type	Action
	EOT sequence or DLE EOT (a switched line disconnect)	Send control sequence and take ?SSND normal return. Calling station can issue ?SSND initial, ?SRCV initial, or ?SDBL (disable line) call now.
All types (point-to-point and multipoint)	Send continue with temporary text delay (?STTD)	Send TTD sequence and wait for NAK reply. - If receiver responds with DLE EOT, then take ?SSND error return (error code ERDIS) and begin BSC error-recovery procedures. - If receiver responds with other than NAK or DLE EOT, take ?SSND error return on error ERUNI and begin BSC error-recovery procedures. - If NAK within time-out interval, take ?SSND normal return.
Multipoint control station	Send initial (with data) (?CINT)	Send select sequence (pointed to by AC0) to receiver and wait for ACK0 response. - If bid successful, ?CINT is changed to ?CONT; otherwise, remains the same. - If no ACK0 within time-out interval, then begin BSC error-recovery procedures. - If ACK0 (acknowledging select sequence), then send data block and wait for ACK1. - If no ACK1 within time-out interval, then begin BSC error-recovery procedures. - If ACK1, then take ?SSND normal return.

(Continued)

Table 2-182. ?SSND Call Types

Station Type	Call Type	Action
	Send continue (with data) (?CONT)	Functionally identical to point-to-point send continue calls.
Multipoint	Send initial (with data)	Not applicable. (You will never issue an ?SSND initial if you are a multipoint tributary; instead, you issue an ?SRCV initial to monitor the line for your poll address. If you are polled and have data to send, issue an ?SSND send continue and send the first block of data.)
	Send continue (with data) (?CONT)	Functionally identical to point-to-point send continue calls.

(Concluded)

Notes

- See the description of ?SRCV in this chapter.

?STMAP

Sets the data channel map.

?STMAP
error return
normal return

Input

AC0 Contains the following:

- Bit 1 is a flag bit:

  Bit 1 = 0 if device is not using a map definition table

  Bit 1 = 1 if device is using a map definition table (defined with ?IDEF)

- Bits 0 and 2 through 31 contain the device code

AC1 Contains the following:

- Bit 0 = 0
- Bits 1 through 31 contain the address of the user's device buffer

AC2 If Bit 1 of AC0 is 1 (map definition table is used), flag bits:

Bits 16–19 = map table entry number in range 0 through 7

Bits 20–25 = offset of first map slot to load in this map table entry

Bits 26–31 number of map slots to load

If Bit 1 of AC0 is 0 (map definition table is not used), AC2 is not used

Output

AC0 Unchanged

AC1 Contains the following:

- Bit 0 = 0
- Bits 1 through 31 contain the buffer's logical address

AC2 Unchanged (32-bit processes only)

High-order portion of buffer address in AC1 (16-bit processes only)

Licensed Material – Property of Data General Corporation

If your program executes as a 16-bit process, change the following bit numbers in AC0, AC1, and AC2.

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

Error Codes in AC0

ERDNM	Illegal device code
ERPRE	System call parameter address error
ERPRV	Caller not privileged for this action
ERPTY	Illegal process type (for example, a 32-bit process tried to set the flag bits in AC0)

Why Use It?

?STMAP works in conjunction with ?IDEF. That is, ?IDEF defines a user device, its I/O channel (BMC or DCH), and optionally, its map slots. ?STMAP initializes the map slots.

What It Does

?STMAP initializes slots in the data channel (DCH) map or burst multiplexor channel (BMC) map for the device that you specify in AC0. ?STMAP also returns the first logical address of the device buffer.

?STMAP Continued

Before you issue ?STMAP, perform the following steps:

1. Set up an I/O buffer for the data transfers in your logical address space. (Note that you must wire the buffer to your working set before you issue ?STMAP. See the description of ?WIRE in this chapter.)

2. Load the buffer address into Bits 1 through 31 of AC1 (Bit 0 must contain 0).

3. Load Bits 2 through 31 of AC0 with the device code you assigned in ?IDEF.

If you did not define a map definition table, all slots that you defined when you issued ?IDEF will be loaded when you issue ?STMAP. The operating system assumes contiguous user logical buffer addresses if you defined more than one slot at ?IDEF time.

If you defined a map definition table, set Bit 1 of AC0 to 1, and set AC2 by performing the following steps:

1. Load Bits 16 through 19 with the number of the map table entry for this device (the entries are numbered 0 through 7).

2. Load Bits 20 through 25 with the offset of the first map slot you want to initialize within the group that this entry allocates.

3. Load Bits 26 through 31 with the total number of map slots (range: 1 through 37 for DCH map; 1 through 1024 for BMC map).

Bits 16 through 17 and 26 through 31 of AC2 correspond to the entry number and total number of map slots you defined in the map definition table (with ?IDEF).

Bits 20 through 25 of AC2 must contain an offset from the device's first map slot. The first map slot is the number the operating system returned to the map definition table entry (First Acceptable Slot field) as output to ?IDEF. Thus, if you want to initialize all the device's map slots, set Bits 20 through 25 to 0. If you want to initialize all slots except the first slot, set Bits 20 through 25 to 1.

Notes

- See the description of ?IDEF in this chapter.

Licensed Material - Property of Data General Corporation

?STOD

Sets the system clock.

?STOD
error return
normal return

Input

AC0	Seconds from 0 through 59
AC1	Minutes from 0 through 59
AC2	Hour from 0 through 23

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERPRV	Caller not privileged for this action (The calling process is not PID 2.)
ERTIM	Illegal time argument (The second, minute, or hour value is outside the legal range.)

Why Use It?

?STOD, which is similar to ?SDAY, allows you (the operator at PID 2) to adjust the system clock at runtime.

What It Does

?STOD sets the system clock to the second, minute, and hour values that you specify in AC0, AC1, and AC2, respectively. Only the operator process (PID 2), or a process that has System Manager privilege, can issue ?STOD.

?STOM

Sets the time–out value for a device.

?STOM
error return
normal return

Input

AC0 One of the following:

- Channel number for the device
- Byte pointer to the name of the device

AC1 Flag bit:

- Bit 0 = 0 if AC0 contains a byte pointer
- Bit 0 = 1 if AC0 contains a channel number

AC2 One of the following:

- Time–out value in seconds
- –1 for the default time–out value

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERICN	Illegal channel
ERIFT	Illegal file type IPC error codes
ERVBP	Invalid byte pointer passed as a system call argument

Licensed Material – Property of Data General Corporation

Why Use It?

You can use ?STOM to set a time-out value, which will prevent the operating system from taking an error return or beginning error-recovery procedures right away. Instead, the operating system will wait for a response from the target device for however long you specify before it assumes that there is a problem with the device. If your time-out expires before the target device responds, the operating system proceeds as if you had not issued ?STOM.

What It Does

?STOM sets a time-out value (in seconds) for the device that you specify in AC0. A time-out value is the length of time the operating system will wait for a response from the target device before it takes an error return or begins error-recovery procedures. If the target device is a character one, the timeout-value applies to individual characters in a response — not to all the characters in a response. The shortest possible time-out value is 2 seconds. If you set the time-out to a value less than 2 seconds, the operating system sets it to 2 seconds. If the device allows input and output, the time-out value that you specify applies to both.

If the target device is a character device, be sure to select the time-out enable characteristic (?CTO) in the device characteristics word to enable time-outs.

If you set AC2 to -1, the operating system uses one of the following default time-out values:

Device	Default Time-out Value
All consoles	-1 (The OS does not check time-out.)
CDR (card reader)	2 seconds
PLT (plotter)	2 seconds

?SUPROC

Enters, leaves, or examines Superprocess mode.

?SUPROC
error return
normal return

Input

AC0 One of the following:

- −1 to turn Superprocess mode on
- +1 to turn Superprocess mode off
- 0 to return the current status of Superprocess mode

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 One of the following:

- −1 if Superprocess mode is on
- +1 if Superprocess mode is off

AC1 Undefined

AC2 Undefined

Error Codes in AC0

ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action (The caller lacks privilege ?PVSP.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can use ?SUPROC to enter or leave Superprocess mode as long as your calling process has privilege ?PVSP. Superprocess mode permits a process to override the operating system's process protections. For example, a process might enter Superprocess mode to terminate (?TERM) or block (?BLKPR) an unrelated process or one of its superiors, such as its father.

Licensed Material – Property of Data General Corporation

Note that a process in Superprocess mode can terminate any other process. Therefore, you should restrict privilege ?PVSP.

In addition, you can use ?SUPROC to find out if your process is currently in Superprocess mode. For this use of ?SUPROC, no privilege is necessary.

What It Does

?SUPROC allows a calling process that has the ?PVSP privilege to enter or leave Superprocess mode. A process that is in Superprocess mode can issue ?BLKPR, ?BRKFL, ?CTYPE, ?PRIPR, ?TERM, or ?UBLPR system calls against any process, not just its subordinate processes. Any process can issue ?SUPROC to determine whether it is currently in Superprocess mode. The calling process does not need privilege ?PVSP to use ?SUPROC for this purpose.

A process remains in Superprocess mode until it terminates or until it issues another ?SUPROC to turn off Superprocess mode.

?SUS

Suspends the calling task.

?SUS
error return
normal return

Input

None

Output

None

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?SUS to suspend the task that is currently executing and transfer control to another task — either the next ready task in the same priority level or the first ready task in the next priority level.

What It Does

?SUS suspends the calling task until another task explicitly readies it with ?IDRDY or ?PRRDY, terminates it with ?IDKIL or ?PRKIL, or redirects it with ?IDGOTO.

Notes

- See the descriptions of ?IDRDY, ?PRRDY, ?IDKIL, ?PRKIL, and ?IDGOTO in this chapter.

Licensed Material - Property of Data General Corporation

?SUSER Enters, leaves, or examines Superuser mode.

?SUSER
error return
normal return

Input

AC0 One of the following:

- −1 to turn Superuser mode on
- +1 to turn Superuser mode off
- 0 to return the current status of Superuser mode

AC1 Reserved (Set to 0.)

AC2 Reserved (Set to 0.)

Output

AC0 One of the following:

- −1 if Superuser mode is on
- +1 if Superuser mode is off

AC1 Undefined

AC2 Undefined

Error Codes in AC0

ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

You can use ?SUSER as you would use ?SUPROC, except that it allows a process unrestricted access to files, rather than to processes. You should restrict the right to enter Superuser mode (privilege ?PVSU).

In addition, you can use ?SUSER to find out if your process is currently in Superuser mode. For this use of ?SUSER, no privilege is necessary.

?SUSER Continued

What It Does

?SUSER allows a process that has the ?PVSU privilege to enter or leave Superuser mode. Superuser mode gives a process the right to access any file on your system, regardless of the file's access controls. A process can also issue ?SUSER to find out whether it is in Superuser mode. The calling process need not have privilege ?PVSU to use ?SUSER for this purpose.

Once a process is in Superuser mode, it remains in that state until it terminates or until it issues another ?SUSER to turn off Superuser mode.

If you disable access controls (in the system-generation procedure), all processes run in Superuser mode. If you then attempt to turn off Superuser mode, the operating system takes the normal return with -1 in AC0.

Notes

- See the description of ?SUPROC in this chapter.

?SYLOG — Manipulates the system log file.

?SYLOG
error return
normal return

Input

AC0 One of the following:

- 1 to start logging events in :SYSLOG
- 0 to stop logging events in :SYSLOG
- −1 to return the current state of the logging function

AC1 One of the following:

- Byte pointer to a name for the system log file
- 0 to retain the current name

AC2 Reserved (Set to 0.)

Output

AC0 One of the following:

- 0 if the logging function is off
- 1 if the logging function is on

AC1 Unchanged

AC2 Undefined

Error Codes in AC0

ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action (The caller is not PID 2.)
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

You can use ?SYLOG to rename the system log file each day. This allows you to keep an ongoing record of system use.

?SYLOG Continued

What It Does

?SYLOG allows the operator process to create or rename the system log file, :SYSLOG, or to determine the current state of the logging function (on or off). Only the operator process (PID 2), or a process with System Manager privilege, can issue ?SYLOG.

The input values of AC0 and AC1 determine which function ?SYLOG performs. If you use ?SYLOG to start the logging function (AC0 = 1) and the system log file does not exist, the operating system creates it.

Once the logging function is turned on, the operating system writes all log file messages to :SYSLOG, including all messages sent by the operating system and/or EXEC, and all messages sent by users via ?LOGEV. The access control list for the :SYSLOG file is OP,OWARE, which gives the operator process full access to the file and shuts out other users. The :SYSLOG file has the special file type ?FLOG.

To close :SYSLOG and prevent the operating system from logging any further events or messages, turn off the logging function (AC0 = 0).

To rename the operating system log file, load AC1 with a byte pointer to the new name before you issue ?SYLOG. The operating system closes the current :SYSLOG file, if it is open, and then renames it.

If you leave the logging function on while you rename the log file, or if you rename the log file and then issue another ?SYLOG to resume logging, the operating system opens a new log file called :SYSLOG and writes all subsequent log messages to that file. All previous log files are strictly historical.

If an error occurs during ?SYLOG, the operating system leaves the current state of the system log file and its name intact.

Notes

- See the description of ?LOGEV in this chapter.

?SYSPRV Enters, leaves, or examines a privilege state.

?SYSPRV *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?SYSPRV packet

Error Codes in AC0

ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action
ERPVM	Unknown privilege mode
ERPVO	Privilege cannot be held exclusively
ERPVP	Other processes using privilege
ERPVX	Privilege held exclusively by other process

Why Use It?

Use this system call to inquire about or change certain process privilege states. You need certain privileges before you can execute some system calls. For example, your process must have System Manager privilege turned on before it can issue a ?JPINIT call to initialize a job processor.

What It Does

This system call reports on or modifies your process's privileges. These privileges — each of which is on or off — are Superuser, Superprocessor, or System Manager.

?SYSPRV Continued

Figure 2–207 shows the structure of the ?SYSPRV parameter packet, and Table 2–183 describes its contents.

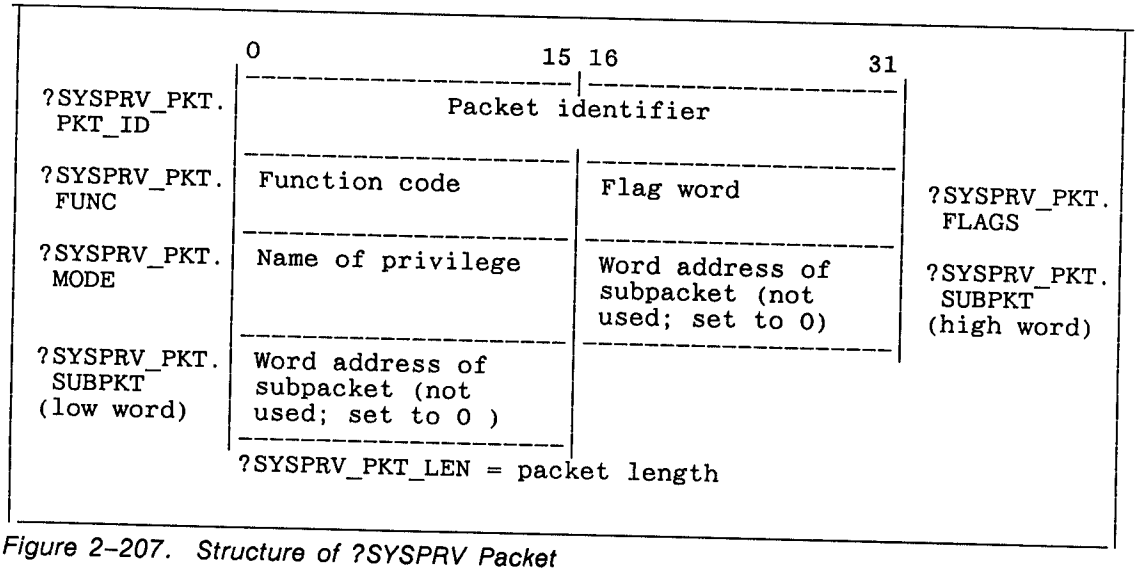


Figure 2–207. Structure of ?SYSPRV Packet

Licensed Material – Property of Data General Corporation

Table 2-183. Contents of ?SYSPRV Packet

Offset	Contents
?SYSPRV_PKT.PKT_ID (double word)	Packet identifier. Place ?SYSPRV_PKT_PKTID here.
?SYSPRV_PKT.FUNC	Function code. The following values have the effects next to them. ?SYSPRV_GET — Get privilege status. ?SYSPRV_ENTER — Enter a privileged mode. ?SYSPRV_ENTER_EXCL — Enter a privileged mode exclusively. ?SYSPRV_LEAVE — Leave a privileged mode.
?SYSPRV_PKT.FLAGS	Flag word. AOS/VS uses it only if offset ?SYSPRV_PKT.FUNC contains ?SYS_PRV_GET. In this case, AOS/VS returns the truth values (1=true, 0=false) of the following bits. ?SYSPRV.FLAGS.CALLER — Caller has privilege on. ?SYSPRV.FLAGS.CALLER_EXCL — Caller has privilege on exclusively. ?SYSPRV.FLAGS.OTHERS — Other processes have privilege on. ?SYSPRV.FLAGS.OTHERS_EXCL — Caller has privilege on exclusively.
?SYSPRV_PKT.MODE	Privilege code. ?SYSPRV_SUSER — Superuser mode. ?SYSPRV_SPROCESS — Superprocess mode. ?SYSPRV_SYSMGR — System manager mode.
?SYSPRV_PKT.SUBPKT (double word)	Word address of subpacket. Currently, there is no subpacket, so set this double word to zero.

Notes

- Next is the correspondence between ?SYSPRV and equivalent system calls:
  - ?SUSER is the equivalent of ?SYSPRV with the Superuser privilege.
  - ?SUPROC is the equivalent of ?SYSPRV with the Superprocess privilege.
  - ?PSTAT is the equivalent of ?SYSPRV with the value ?SYSPRV_GET in offset ?SYSPRV_PKT.FUNC.
- A process remains in a privileged state until it terminates or until it turns that state off with another ?SYSPRV call.
- The system log records each execution of ?SYSPRV.

?TASK

Initiates one or more tasks.

?TASK *[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 (TDP), unless you specify the address as an argument to ?TASK

Output

AC0	Undefined
AC1	Undefined
AC2	Address of the task definition packet (TDP)

Error Codes in AC0

ERNOT	No task control block available
ERSTS	Stack too small
ERTID	Task ID error
ERTMT	Too many tasks requested
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?TASK allows you to create new tasks. By creating multiple tasks, you can take full advantage of the operating system's multitasking environment.

What It Does

?TASK initiates one or more tasks to create a multitasking environment. If the ?TASK caller specifies a starting PC within Ring 7, then control passes to a task initiation routine called ?UTSK. This occurs during the initiation sequence invoked by ?TASK.

Licensed Material - Property of Data General Corporation

By default, the operating system uses a dummy ?UTSK routine in the user runtime library URT32.LB. This routine returns control immediately to the task scheduler, which initiates the tasks. As an alternative, you can supply your own ?UTSK task initiation routine and link it with your program. (See "Task Initiation" for more information on task initiation routines.)

When the operating system starts executing a new task, it loads AC3 with a pointer to the system task termination code. You should begin the code for each task (except the initial task) with a WSSVS or a WSSVR instruction, and end the code with a WRTN instruction. Do not use the WSSVS/WSSVR and WRTN sequence for the initial task.

Each ?TASK requires a task definition packet, which you must define in your logical address space. There are two packet formats: a standard format and an extended format for queued task creation. The symbol ?DSLTH represents the standard packet's length. Figure 2–208 shows the structure of the standard task definition packet and Table 2–184 describes the contents of each offset.

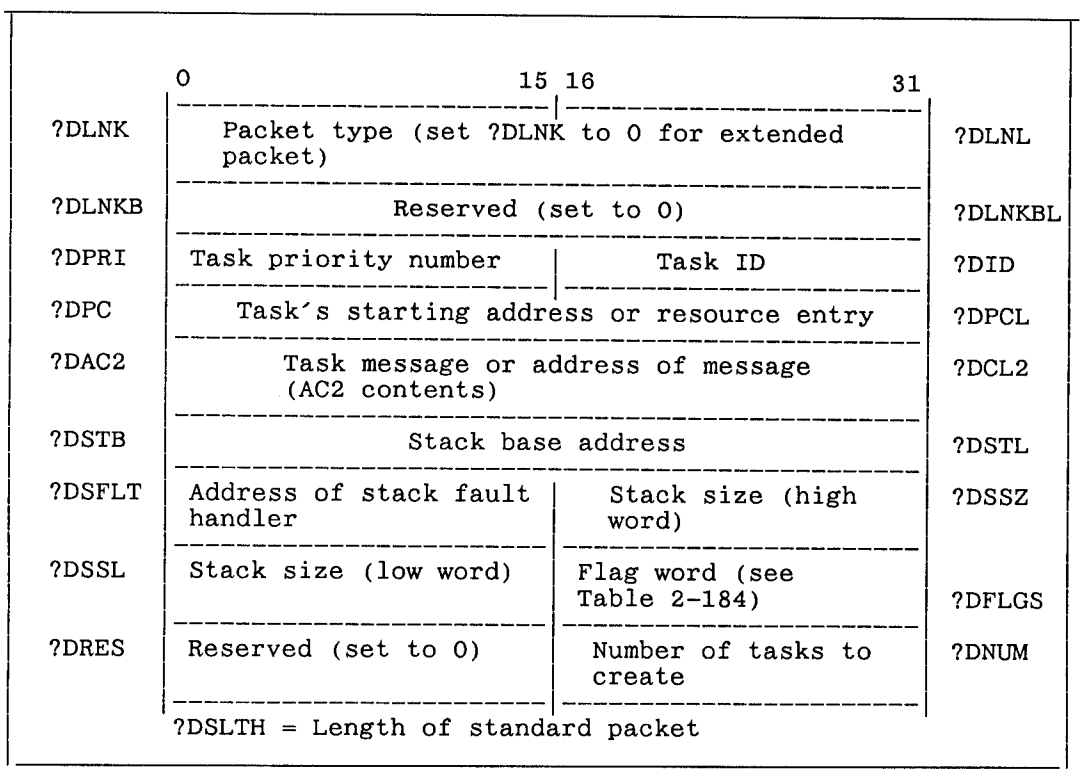


Figure 2–208. Structure of Standard Task Definition Packet

Table 2-184. Contents of Standard Task Definition Packet

Offset	Contents
?DLNK (double word)	Packet type. Set to 0 for the extended packet.
?DLNKB (double word)	Reserved. (Set to 0.)
?DPRI	Task priority number. Set to 0 for same priority as calling task.
?DID	Task ID. Set to 0 for no priority.
?DPC (double word)	Starting address of task or resource entry.
?DAC2 (double word)	Task message or address of message (initial AC2 contents.)
?DSTB (double word)	Stack base address. Set to -1 for no stack.
?DSFLT	Address of stack fault handler. Set to -1 for the current address.
?DSSZ	Stack size (high word.) If there is no stack, the O S ignores this offset.
?DSSL	Stack size (low word).
?DFLGS	Task flag word. ?DFLRC--Resource call task.
?DRES	Reserved. (Set to 0.)
?DNUM	Number of tasks you want to create.

Offset ?DLNK defines the packet type. If you are using the extended packet, set the one-word field ?DLNK to 0. If you select the standard packet, ?DLNK can contain any nonzero value.

Set offset ?DNUM to the number of tasks you are creating with ?TASK. You can define as many as 32 potential tasks per program, either at assembly time or at link time.

Licensed Material - Property of Data General Corporation

Offset ?DPRI contains the priority number of the newly created task or tasks. If you default this parameter by setting it to 0, the operating system assigns the new task(s) the same priority as the calling task. The packet contains only one parameter for task priority. Therefore, if you use a single ?TASK to initiate more than one task, the operating system initiates all of the new tasks at the same priority level.

Offset ?DID contains the TID of the new task. Do not set ?DID to a TID that is already in use.

If you are using ?TASK to initiate more than one task, the value of ?DID becomes the TID for the first task and the base for the other TIDs. That is, the operating system increments the ?DID value by 1 for each additional TID. When you default this parameter (to 0), the operating system does not assign TIDs to the new tasks.

Offset ?DPC contains the starting address of the task code; that is, the location where execution will begin. This can be an address within your unshared logical context or an address within a shared routine.

You can use the task definition packet to send a 32-bit message, via AC2, to each new task as soon as it assumes control. If you are initiating only one task with ?TASK, place the message in offset ?DAC2. If you are initiating more than one task, offset ?DAC2 should point to a message table that contains one 32-bit message for each task. If you do not want to send task messages, set offset ?DAC2 to 0.

Offset ?DSTB specifies the stack base; that is, the starting address of the new task's stack. Offset ?DSSZ indicates the stack size, which must be at least 60 words for a 32-bit program. If you set ?DSTB to -1 (the default), which specifies no stack, you must allocate the stack after the task begins executing.

Only the Ring 7 stack base addresses can be supplied to ?TASK. This is true even if ?TASK is issued from a ring other than Ring 7. (See the description of the ?RINGLD system call in this chapter for more information on stack support in rings other than Ring 7.)

If you are initiating more than one task, you must define a stack at least as large as ?DNUM*?DSSZ; that is, the total number of tasks (?DNUM) times the stack size. The operating system builds the stacks contiguously, offset from the same stack base. Figure 2-209 compares the stacks generated for single-task ?TASK system calls and multitask ?TASK system calls.

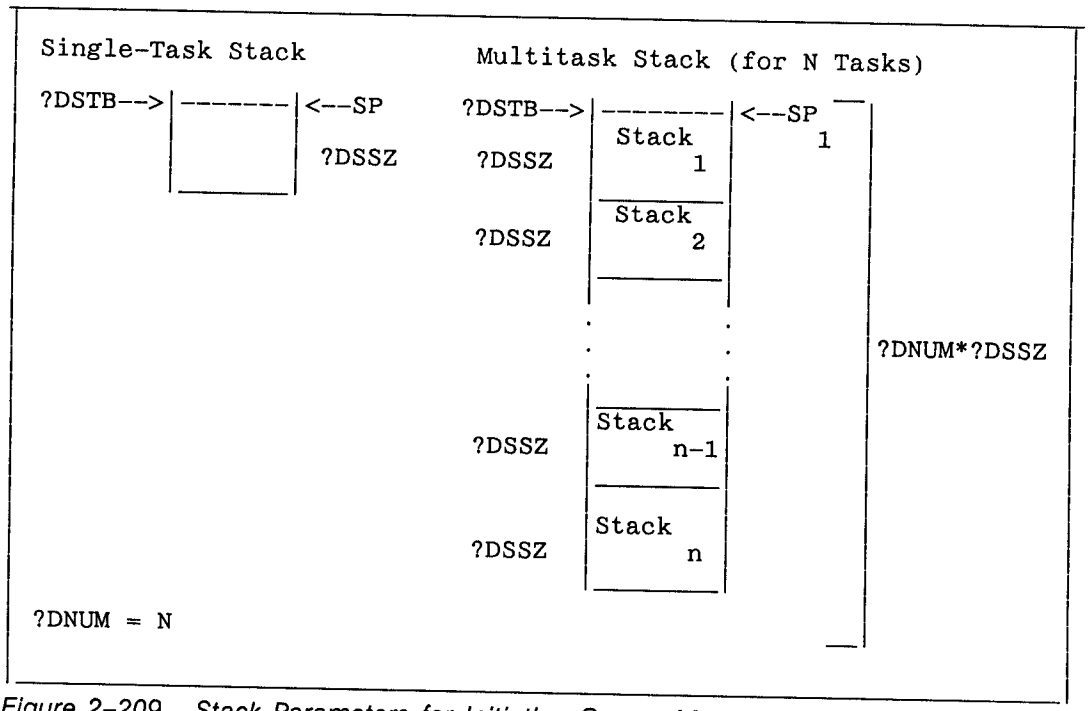


Figure 2-209. Stack Parameters for Initiating One or More Tasks

Offset ?DSFLT points to the address of the stack fault handler. The stack fault handler is a routine that gains control when there is a stack error. You can define your own stack fault handler or you can use the default stack fault handler in URT32.LB. To use the default stack fault handler, set ?DSFLT to -1. ?DSFLT is a 16-bit value. Therefore, the stack fault handler must be in the first 64K words of the ring.

Extended Task Definition Packet

The ?TASK extended packet contains all the parameters specified in the standard task packet, plus additional information for the queued task creation option. The extended packet is ?DXLTH words long. Figure 2-210 shows the structure of the extended packet, and Table 2-185 describes the contents of each offset.

Before you initiate queued tasks, issue the ?IQTSK system call, which creates a queued task manager to handle the queue. The ?DQTSK system call removes one or more task definition packets from a task creation queue.

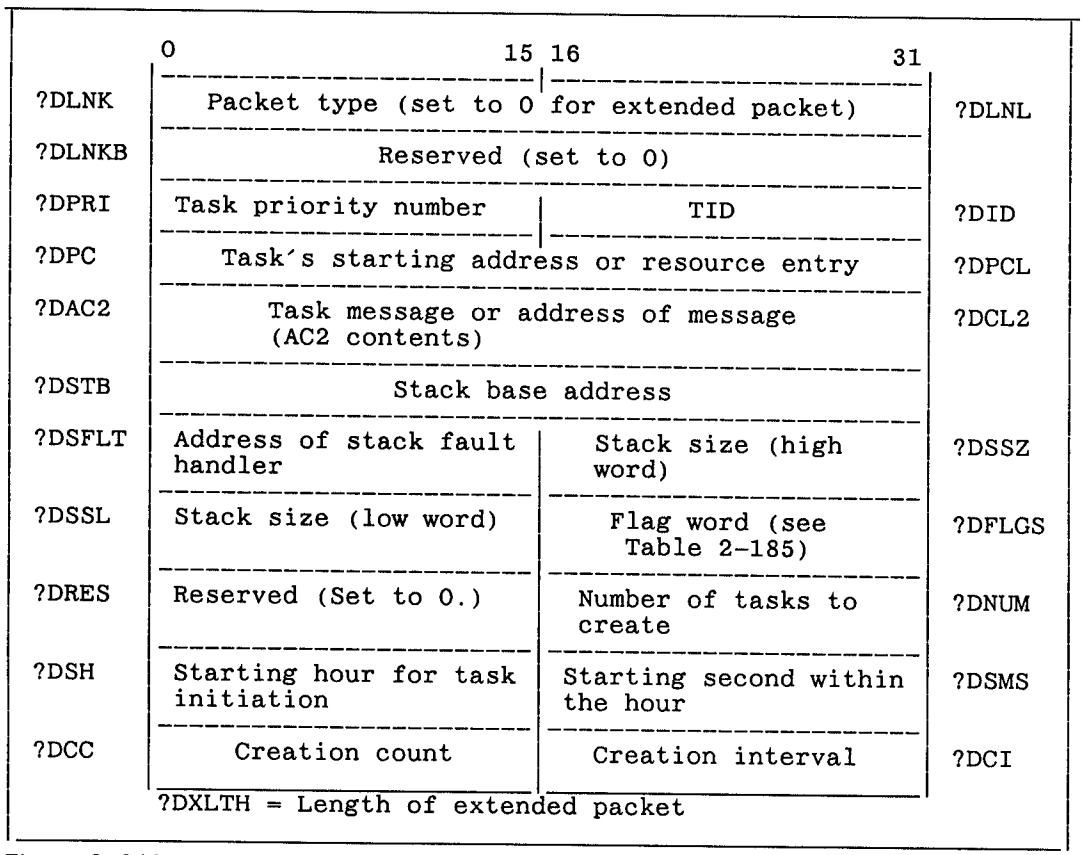


Figure 2-210. Extended Task Definition Packet

Offset ?DSH, starting hour, specifies the hour at which the operating system will begin initiating the new tasks. If you set this parameter to −1, the default, the operating system initiates the new tasks immediately and ignores the ?DSMS parameter.

Offset ?DSMS tells the operating system which second within the hour to start initiating the tasks. The range for this parameter is 1 through 3600 (there is no default value).

Table 2-185. Contents of Extended Task Definition Packet*

Offset	Contents
?DLNK (double word)	Packet type. Set nonzero value for standard packet or set to 0 for extended packet.
?DLNKB (double word)	Reserved. (Set to 0.)
?DPRI	Task priority number. Set to 0 for same priority as calling task.
?DID	TID. Set to 0 for no priority.
?DPC (double word)	Starting address of task or resource entry.
?DAC2 (double word)	Task message or address of message (initial AC2 contents.)
?DSTB (double word)	Stack base address. Set to -1 for no stack.
?DSFLT	Address of stack fault handler. Set to -1 for current address.
?DSSZ	Stack size (high word). The OS ignores this if there is no stack.
?DSSL	Stack size (low word).
?DFLGS	Task flag word. ?DFLRC--Resource call task.
?DRES	Reserved. (Set to 0.)
?DNUM	Number of tasks you want to create.
?DSH	Starting hour for task initiation. Set to -1 to initiate the task immediately.
?DSMS	Starting second within the hour for task initiation. (The OS ignores this offset if you set ?DSH to -1 for immediate task initiation.)
?DCC	Number of times to initiate the task(s).
?DCI	Amount of time between initiating tasks.

* There is no default unless otherwise specified.

Licensed Material - Property of Data General Corporation

Offset ?DCC specifies the number of times you want the system to repeat the task initiation(s). Set this parameter to −1 if you want the system to initiate the tasks repeatedly at the time intervals defined in ?DCI. Set ?DCC to 1 if you want to initiate the task(s) only once.

Offset ?DCI specifies the creation interval; that is, the number of seconds between each reinitiation. There is no default value for this parameter.

Sample Packet

The following sample packet creates three tasks:

```
PKT:     .BLK    ?DSLTH          ;Allocate enough space for this
                                 ;packet.  Packet length = ?DSLTH.
         .LOC    PKT+?DLNK       ;Specify extended or standard packet.
         .WORD   1               ;Standard packet (can contain any
                                 ;nonzero number).
         .LOC    PKT+?DLNKB      ;Reserved.
         .WORD   0               ;You must set this value to 0.
         .LOC    PKT+?DPRI       ;Specify priority of new tasks.
        .WORD    2               ;Medium priority. (Specify 0 for same
                                 ;priority as calling task.)
         .LOC    PKT+?DID        ;Specify TID of first task.
         .WORD   2               ;TID is 2 for first task. (The OS
                                 ;increments the TID of each
                                 ;additional task by 1.)
         .LOC    PKT+?DPC        ;Specify task's starting address.
         .DWORD  STRT            ;Starting address of each new task is
                                 ;STRT.  The new tasks execute the
                                 ;same code.
         .LOC    PKT+?DAC2       ;Specify any task messages.
         .DWORD  MESTAB          ;MESTAB is a table that contains a
                                 ;different message in AC2 for each
                                 ;task.
         .LOC    PKT+?DSTB       ;Stack base address.
         .DWORD  STKBAS          ;Stack base address is STKBAS.
         .LOC    PKT+?DSSZ       ;Size of stack for each task.
         .DWORD  STKSZ           ;Stack size is STKSZ.
         .LOC    PKT+?DSFLT      ;Address of stack fault handler.
         .WORD   -1              ;Use the OS default stack handler,
                                 ;URT32.LB.
         .LOC    PKT+?DFLGS      ;Task flag word.
         .WORD   0               ;There is no task flag word.
         .LOC    PKT+?DRES       ;Reserved.
         .WORD   0               ;You must set this value to 0.
         .LOC    PKT+?DNUM       ;Number of tasks to start?
         .WORD   3               ;Start 3 tasks.
         .LOC    PKT+?DSLTH      ;End of packet.
STKBAS:  .BLK    STKSZ*3         ;When you declare the stack base
                                 ;address for the first task, the OS
                                 ;divides it by the number of tasks
                                 ;you are starting.
```

Notes

- See the description of ?RINGLD in this chapter for information on stack support in rings other than Ring 7.
- See the descriptions ?IQTSK and ?DQTSK in this chapter.

Licensed Material - Property of Data General Corporation

?TERM

Terminates a process.

?TERM
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the name of the process you want to terminate
- PID of the process you want to terminate
- −1 to terminate the calling process

AC1 One of the following:

- −1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID

Otherwise, the contents of AC1 are ignored.

AC2 One of the following:

- Word address of the IPC message header
- 0 if there is no message header

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

NOTE: AC2 input values are valid only if the calling process issues ?TERM to terminate itself.

?TERM Continued

Error Codes in AC0

ERMPR	System call address error
ERPNM	Illegal process name format
ERPRE	Invalid system call input, AC0 is not −1 and AC1 is not 0 or −1
ERPRH	Attempt to access a process not in the hierarchy
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?TERM allows you to explicitly terminate a process. Unless ?RETURN or ?TERM terminate a process, the process continues to exist until either the last task in the process kills itself, or the process traps because of a programming or system error.

What It Does

A process issues ?TERM to terminate itself or any subordinate process, or, if it is in Superprocess mode, a process can use ?TERM to terminate any other process.

When a process terminates, so do all its offspring (sons, grandsons, and so forth). A process can terminate a subordinate even if the subordinate has a higher priority number or is a different process type.

If a process terminates itself, the operating system sends its father an IPC termination message. This IPC termination message is either a standard termination message or the message that you specify in AC2. In the latter case, the operating system ignores the ports designated in the IPC header.

Notes

- See the description of ?RETURN in this chapter.

?TIDSTAT

Returns status of target task. (32–Bit Processes Only)

?TIDSTAT
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	TID of task whose status you want to investigate
AC2	Reserved (Set to 0.)

Output

AC0	Status word
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERTID	Task with specified task ID (TID) not found
ERICM	16–bit program incorrectly attempted to call ?TIDSTAT

Why Use It?

You can use ?TIDSTAT to find out what is happening to a particular task.

What It Does

?TIDSTAT returns a word in AC0 that describes the target task's status. (Refer to PARU.32 and PARU_LONG for a complete list of the currently defined status words.)

?TLOCK — Protects a task from being redirected.

?TLOCK
error return
normal return

Input

AC0 ?TMYRING flag. If set, locks caller against task–redirection system calls issued from its own ring

AC1 Reserved (Set to 0.)

AC2 One of the following:

- Address of a double–word mailbox to contain notification of a pended redirection attempt
- –1 to indicate that the caller does not want to specify a mailbox

Output

AC0 Flag bits:

- ?TALOCK — Ring was already issued for this task when ?TLOCK was issued
- ?TMYRING — Ring was already locked against task–redirection system calls issued from caller's ring (This bit is only set if ?TALOCK is also set.)

AC1 Undefined

AC2 Address of the old mailbox (if any)

Error Codes in AC0

No error codes are currently defined.

Licensed Material – Property of Data General Corporation

Why Use It?

?TLOCK allows you to protect inner-ring segments from being redirected by ?IDGOTO, ?IDKIL, ?PRIKIL, ?IDSUS, and ?PRSUS task-redirection system calls issued from its own or higher rings.

What It Does

?TLOCK accepts an input argument in AC2 that specifies the address of a double-word mailbox. If you provide a mailbox address, the operating system can inform a locked task that another task is trying to redirect it. Then, the operating system places a nonzero flag in the mailbox when a task-redirection request by another task pends.

A task that is executing within an inner-ring segment image can poll the mailbox, decide whether to abort its progress in the critical region, and then issue ?TUNLOCK to allow the redirection request to proceed.

?TLOCK can lock a task against being redirected by task-redirection system calls issued from its own ring. To do this, the caller must specify the ?TMYRING flag in AC0 on input.

The operating system can set the ?TMYRING flag in AC0 on output. In this case, the operating system sets ?TMYRING if the caller's ring was already locked against task-redirection system calls issued from within the same ring specified at ?TLOCK input. However, the operating system sets ?TMYRING on output only if ?TALOCK is also set.

Successive ?TLOCK system calls can change the state of task-redirection protection. When ?TLOCK changes the address of the mailbox associated with the calling task in the ring, the operating system places the current contents of the old mailbox in the new mailbox. Each successive ?TLOCK outputs the values of the state variables as they existed before the task issued ?TLOCK. Every task that uses ?TLOCK must save the values of these state variables until it leaves its critical region. Then, after it has left its critical region, each task must appropriately restore the state variables.

?TLOCK allows callers to retain task-redirection protection while changing either the mailbox address or the ?TMYRING variable. To do this, however, the caller's initial ?TLOCK must indicate that task-redirection protection was not already in effect when the task entered its critical region.

Consecutive ?TLOCKs from the same ring on behalf of a single task are legal, but they do not provide further protection.

To cancel the effect of a ?TLOCK, a task must issue ?TUNLOCK.

Notes

- See the description of the ?TUNLOCK system call in this chapter.

?TMSG Defines the termination message format.

?TMSG
error return
normal return

Input

AC0	Format revision number
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERRUN	Revision number is unknown to this version of the operating system

Why Use It?

?TMSG defines the format of termination messages that your process expects to receive.

What It Does

?TMSG applies only to issuing processes that are type B or type C. The process is either a 16-bit one or a 32-bit one. Figure 2-211 describes the format of the termination message that a 32-bit process receives, and Table 2-186 describes its contents. Figure 2-212 describes the format of the termination message that a 16-bit process receives and Table 2-187 describes its contents.

In either case (the calling process is 32- or 16-bit), word ?IUFL of the termination IPC header contains the value of ?TEXT in the termination field. This value indicates that the termination message is an extended one. 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.

Licensed Material - Property of Data General Corporation

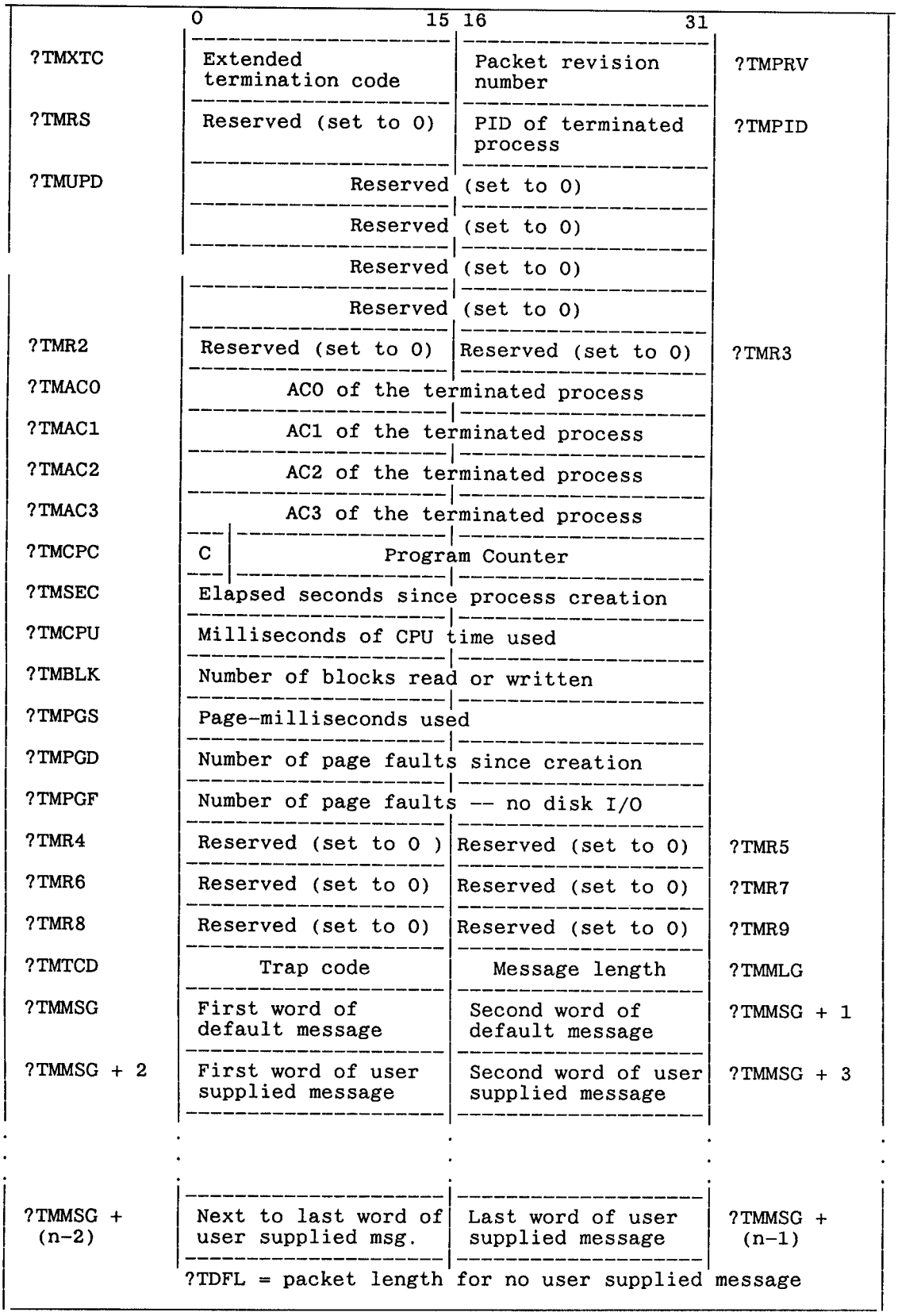


Figure 2-211. Structure of Termination Message a 32-bit Process Receives

Table 2-186. Contents of Termination Message a 32-bit Process Receives

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 console 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.

(Continues)

Table 2-186. Contents of Termination Message a 32-bit Process Receives

Offset	Contents
?TMCPC (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)

	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)
?TMOL	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)

Figure 2-212. Structure of Termination Message a 16-bit Process Receives (Continues)

	0 15
?TMBH	Number of blocks read or written (high order)
?TMBL	Number of blocks read or written (low order)
?TMGH	Page-milliseconds used (high order)
?TMGL	Page-milliseconds used (low order)
?TMDH	Page faults since creation (high order)
?TMDL	Page faults since creation (low order)
?TMFH	Page faults -- no disk I/O (high order)
?TMFL	Page faults -- no disk I/O (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	Trap code
?TMMLG	Message length
?TMMSG	First word of default message
?TMMSG + 1	Second word of default message
?TMMSG + 2	First word of user-supplied message
?TMMSG + 3	Second word of user-supplied message
.	. .
.	. .
.	. .
?TMMSG + (n-2)	Next to last word of user-supplied message
?TMMSG + (n-1)	Last word of user-supplied message

?TDFL = packet length for no user supplied message

Figure 2-212. Structure of Termination Message a 16-bit Process Receives (Concluded)

Table 2-187. Contents of Termination Message a 16-bit Process Receives

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 console 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.)
?TMOH	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.
?TMOL	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).

(Continues)

Table 2-187. Contents of Termination Message a 16-bit Process Receives

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 ?XTR16 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)

Table 2-187. Contents of Termination Message a 16-bit Process Receives

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.
?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)

Licensed Material - Property of Data General Corporation

?TPID

Translates a PID.

?TPID
error return
normal return

Input

AC0	PID or virtual PID
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Local PID
AC1	One of the following: • 0 if input PID is local PID • Host ID in Bits 17 through 31 if input PID is virtual PID • −1 if PID is remote and host ID is 0
AC2	Unchanged

Error Codes in AC0

ERPRE	Invalid system call parameter (You supplied an invalid PID in AC0.)

Why Use It?

?TPID decodes a PID into a local PID and a host ID. From this information, you can determine the process and host name to which the virtual PID refers.

What It Does

?TPID translates a virtual PID into its component parts: a local PID and a host ID. ?TPID also tells the caller when the input parameter is strictly a local PID. If ?TPID returns 0 in AC1, the input to AC0 was a local PID.

Licensed Material - Property of Data General Corporation

?TPORT

Translates a local port number to its global equivalent.

?TPORT
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	Caller's local port number

Output

AC0	Undefined
AC1	32-bit global equivalent of the port number in AC2
AC2	Unchanged

Error Codes in AC0

ERIVP	Invalid port number

Why Use It?

A process can issue ?TPORT to determine the global port number assigned to one of its local ports. This information can be useful if you are sending and receiving over multiple ports or if you intend to issue an ?IS.R system call.

What It Does

?TPORT returns the global equivalent of the local port number you specify in AC2. The local port must belong to the ?TPORT caller.

Notes

- See the description of ?IS.R in this chapter.

Licensed Material - Property of Data General Corporation

?TRCON

Reads a task message from the process console.

?TRCON
error return
normal return

Input

AC0	Byte pointer to the message area (which must be word-aligned)
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Message byte count (including the NEW LINE terminator, excluding the TID and commas)
AC2	Undefined

Error Codes in AC0

ERNOT	No task control block available
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?TRCON allows you to pass input from the process console to individual tasks in a multitasking process.

What It Does

?TRCON prepares the calling task to receive a message that you type at your console. ?TRCON is incompatible with ?READ. Therefore, if you use ?TRCON to receive a message, you cannot issue a simultaneous ?READ against that console.

More than one task in a program can issue ?TRCON, because the operating system uses task identifiers (TIDs) to differentiate between the receivers. To use ?TRCON, a task must have a TID.

When you issue ?TRCON, the operating system initiates a message-manager task to monitor the process console for ?TRCON messages. The operating system terminates the message manager if it is still active after all active ready tasks in the process have terminated. (The message manager counts as one of the 32 possible tasks in the program.)

When the message manager is ready to receive a message, the operating system prints a > prompt on the process console. ?TRCON returns normally when the calling process receives the message.

Use the following format for the task messages:

<task-ID><comma>message<NEW LINE>

where:

- task-ID is the TID (in decimal) of the receiving task .
- comma is a required comma (,) as a delimiter.
- message is the message string.
- NEW LINE is a required NEW LINE terminator.

The message string can be up to 132 characters long, including the NEW LINE terminator.

If there is an input error, the operating system returns one or more of the following messages to the process console:

- ***TID NOT FOUND*

  No task with the specified TID issued ?TRCON to receive the message.

- ***INPUT ERROR*

  You used an alphabetic character in the TID

- ***NO TASK WAITING*

  No task has issued ?TRCON

Notes

- See the description of ?READ in this chapter.

?TRUNCATE Truncates a file at the current position.

?TRUNCATE
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Channel number
AC2	Reserved (Set to 0.)

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERIFT Illegal file type

File system error codes

Why Use It?

?TRUNCATE allows you to control the size of a disk or magnetic tape file that is currently open for I/O. For disk files, for example, ?TRUNCATE allows the operating system to reclaim the disk storage that you no longer require.

What It Does

?TRUNCATE truncates disk or magnetic tape files at the file pointer's current position. (The ?TRUNCATE caller must have previously issued ?OPEN to open the selected file.) There is no packet associated with ?TRUNCATE.

When you issue ?TRUNCATE against an open magnetic tape file, the operating system does not actually truncate the file. Instead, to indicate the end of the file, it writes two end-of-file marks (EOF) after the file pointer's position.

If the caller passes a channel number of an open channel that indicates the null file type (@NULL), ?TRUNCATE produces no action, but returns successfully. If the caller passes a channel number that indicates a file type other than any of the file types mentioned above, ?TRUNCATE returns error code ERIFT.

Notes

- See the description of ?OPEN in this chapter.

?TUNLOCK

Allows a task to be redirected.

?TUNLOCK
error return
normal return

Input

AC0 One of the following:

- 0 to remove all protection from this ring
- ?TALOCK to remove protection from my ring only
- ?TALOCK+?TMYRING is a no-operation

AC1 Reserved (Set to 0.)

AC2 One of the following:

- Address of a double-word mailbox to contain the output of the previous ?TLOCK or ?TUNLOCK system call
- -1 to indicate that the caller does not want to specify a mailbox
- 0 to indicate that all protection is being removed

Output

AC0 Unchanged

AC1 Unchanged

AC2 Unchanged

Error Codes in AC0

ERICM Illegal command
ERTLK No matching ?TLOCK for this ?TUNLOCK

Licensed Material - Property of Data General Corporation

Why Use It?

You issue ?TUNLOCK to allow a task-redirection request against a locked task to proceed — provided that the appropriate protection is released.

What It Does

?TUNLOCK removes task redirection protection for the current task in the current ring. It restores redirection protection to a previous state of protection that is equal to or less than the present state of protection. A ?TUNLOCK call basically cancels the effect of an earlier ?TLOCK call on behalf of the same task in the same ring.

Notes

- See the description of the ?TLOCK system call in this chapter.

?UBLPR

Unblocks a process.

?UBLPR
error return
normal return

Input

AC0 One of the following:

- Byte pointer to the name of the target process
- PID of the target process

AC1 One of the following:

- -1 if AC0 contains a byte pointer
- 0 if AC0 contains a PID

AC2 Reserved (Set to 0.)

Output

AC0 Unchanged

AC1 Unchanged

AC2 Undefined

Error Codes in AC0

ERMPR	System call address error
ERPNM	Illegal process name
ERPRE	Invalid system call input (AC1 is not a 0 or a -1)
ERPRH	Attempt to access a process not in the hierarchy
ERVBP	Invalid byte pointer passed as a system call argument

Why Use It?

?UBLPR is the complement of ?BLKPR; thus, if a process was blocked with ?BLKPR, you must use ?UBLPR to unblock it.

?UBLPR is one of the system calls that a process can issue in Superprocess mode.

Licensed Material - Property of Data General Corporation

What It Does

?UBLPR unblocks the target process that you specify in AC0. Note that the target process must have been blocked with the ?BLKPR system call. (To use ?BLKPR against a process other than the caller's process, Superprocess mode must be turned on.)

?UBLPR lifts the suspension imposed by ?BLKPR on all tasks within the target process. It has no effect upon the state of any other outstanding suspensions against those tasks. However, ?UBLPR only unblocks the target process if it has at least one task that is ready to run. Otherwise, the target process remains blocked until one of its tasks becomes ready to run.

Notes

- See the description of ?BLKPR in this chapter.

?UIDSTAT

Returns the status of a task and an unambiguous identifier.

?UIDSTAT
error return
normal return

Input

AC0	Reserved (Set to 0.)
AC1	One of the following: • TID of the specified task (0 is illegal) • −1 to indicate the current task
AC2	Address of the return packet

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

No error codes are currently defined.

Why Use It?

You can use ?UIDSTAT to get a unique task ID, which you can then use to index into multiple-task databases.

Licensed Material - Property of Data General Corporation

What It Does

?UIDSTAT is functionally similar to the ?TIDSTAT and ?IDSTAT system calls; however, unlike ?TIDSTAT and ?IDSTAT, ?UIDSTAT returns a unique TID.

?UIDSTAT accepts a standard TID as input in AC1. However, it also accepts -1 in AC1 to indicate the return of task information for the current task. The operating system passes a word pointer to a return packet in AC2. This packet returns the unique TID, the task status word, the standard TID, and the task's priority. (See Figure 2-213 for the packet structure.)

The unique TID value ranges between 1 and
([maximum number of tasks specified by the Ring 7 process] - 1)

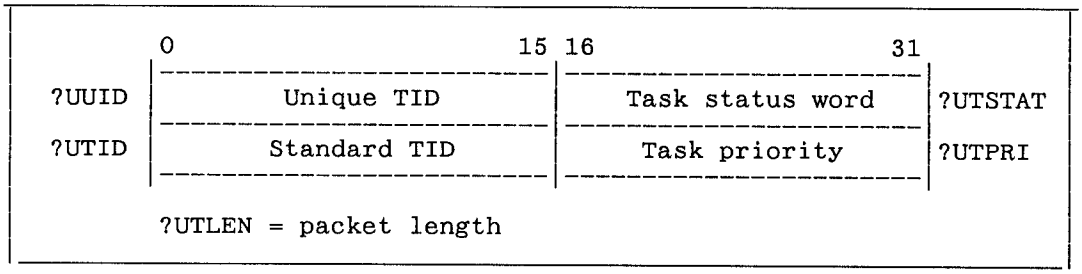


Figure 2-213. Structure of ?UIDSTAT Packet

?UNWIND Unwinds the stack and restores the previous environment (16–Bit Processes Only)

?UNWIND
normal return

Input

None

Output

AC0 One of the following:

- Base address of the restored resource
- 0

AC1 Return address of the previous resource

AC2 Unchanged

Error Codes in AC0

No error codes are currently defined

Why Use It?

?UNWIND pops the current return block off the stack, just as RTN does, but gives you the option of passing control to an address relative to the return address or to an address relative to the base of the restored resource. Thus, you can use ?UNWIND to restore the stack environment of a previously called resource, without necessarily returning to its RTN address.

What It Does

?UNWIND unwinds the user stack and restores it to the previous environment. As output to ?UNWIND, the operating system returns the base address of the restored resource, immediately following the ?UNWIND; that is, control returns to the calling resource. At this point, the calling resource can transfer control to the return address in AC1, to a fixed address relative to the return address (contents of AC1 + n), or to a fixed address within the previous resource (contents of AC0 + n). If you did not use a resource system call to call the target resource, the operating system returns 0 to AC0.

The ?UNWIND caller must be part of the root resource or part of a resource that will not be released when the stack is unwound. Note that the code generated by ?UNWIND cannot be a single word. (This can affect any skip instructions you issue before you issue ?UNWIND.)

Licensed Material – Property of Data General Corporation

?UNWIRE — Unwires pages previously wired.

?UNWIRE
error return
normal return

Input

AC0	Starting address of the region to unwire
AC1	Ending address of the region to unwire
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERICM	Illegal system command (invalid system call for 16-bit processes)
ERPRE	Invalid system call parameter
ERVWP	Illegal system command

Why Use It?

?UNWIRE is the complement of ?WIRE; thus, if a page was wired with ?WIRE, you must use ?UNWIRE to unwire it. You issue one ?UNWIRE to revoke all previous ?WIREs, even though each ?WIRE may be for a different address region.

What It Does

?UNWIRE removes the wired status from a range of addresses in your working set. (A "wired" page is a page that is permanently bound to the working set.) Note that ?UNWIRE can reduce your minimum working set size.

Before you issue ?UNWIRE, load AC0 and AC1 with the starting and ending addresses, respectively, of the region you want to unwire. If the starting and ending addresses fall on page boundaries, ?UNWIRE affects those boundaries and all pages between them. Otherwise, ?UNWIRE affects only the full pages between the starting and ending addresses.

Notes

- See the description of ?WIRE in this chapter.

?UPDATE — Flushes file descriptor information.

?UPDATE
error return
normal return

Input

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

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERACU	Channel not open
ERCIU	Channel in use
ERSIM	Simultaneous request on same channel

Why Use It?

?UPDATE is useful whenever you cannot afford to lose file descriptor information if your system crashes.

What It Does

?UPDATE flushes any memory-resident file descriptor information for the target file to disk, and thereby updates it. File descriptor information includes user data areas (UDAs). The file you update must be a user file; if you attempt to update a system file, ?UPDATE takes the normal return, but has no effect.

?UPDATE guarantees the integrity of all the data you previously wrote to the file via a ?FLUSH system call. Some data, however, such as shared pages and any data buffered by the Agent, may not have been written to disk yet. To flush this data, use the ?ESFF system call or the ?WRITE system call with the force output bit set.

Notes

- See the descriptions of ?WRITE, ?FLUSH, ?ESFF, and ?RPAGE in this chapter.

Licensed Material - Property of Data General Corporation

?VALAD

Validates a logical address.

?VALAD
error return
normal return

Input

AC0	Logical address to validate
AC1	Type of validation
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Indicator of which validations succeeded
AC2	Undefined

Error Codes in AC0 and AC1

ERACC Access denied

Why Use It?

?VALAD allows the caller to verify that an address is valid or that access is available to that address.

What It Does

?VALAD validates a logical address for a particular type of access. You can check ?RDAC (read), ?WRAC (write), ?EXAC (execute), and ?VLAC (validate) access. Also, you can check any combination of access types at one time.

?VALAD takes the error return if any of the requested accesses being checked fails. To indicate which accesses succeeded, the operating system returns the ERACC (access denied) error code in AC0 and AC1.

The logical AND of the access privileges that you want to check and the actual privileges that the address has always return in AC1. Therefore, if the operating system takes a normal return, AC1 appears unchanged. Every privilege does not have to fail for the operating system to take the error return. If there is an error, the privileges that succeed are indicated in AC1.

NOTE: To use ?VALAD optimally, the server must perform the following checks before it issues this system call:

1. Check whether the address is being passed below the ring in which the caller resides (that is, is it a Trojan pointer). If so, fail. If not, succeed.

2. Do a LPHY on the pertinent logical address. If the LPHY succeeds, examine the PTE for the appropriate access. If the PTE does not have the appropriate access, fail. If the PTE does have the appropriate access, succeed.

?VALIDATE Validates an area for Read or Write access.

?VALIDATE *[packet address]*
error return
normal return

Input

AC0	Unused
AC1	Unused
AC2	Address of packet

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

(See also the section "Error Conditions.")

ERACC	Access denied
ERICM	Illegal system command
ERPRE	Invalid system call parameter
ERUFR	Unknown function request
ERVWP	Invalid address passed as system call argument

Why Use It?

?VALIDATE is most useful to inner ring server development. Using the ?VALIDATE call, servers can avoid user traps caused by referencing an invalid area passed from the customer program. See examples in the later section "Examples of Use".

What It Does

?VALIDATE provides the caller with an interface to the Agent validation routines. An area is defined by a word pointer and word length, or by a byte pointer and byte length. Areas can be validated for Read or Write access. Also, there is a validation function for validating strings of unknown length but known maximum length, using a user-specified delimiter table.

Licensed Material - Property of Data General Corporation

Packet Format

The five functions offered by this system call share the same packet format. The type of processing which will occur depends on the contents of offset ?VFUNC. Table 2-188 explains the functions and their codes.

Table 2-188. ?VALIDATE Functions and Their Codes

Offset	Description
?VFUNC	FUNCTION CODES and FLAG ?RVWPL--validate an area defined by word pointer and word length for Read access. ?WVWPL--validate an area defined by word pointer and word length for Write access. ?RVBPL--validate an area defined by byte pointer and byte length for Read access ?WVBPL--validate an area defined by byte pointer and byte length for Write access. ?RVBPX--validate an area defined by byte pointer and maximum byte length (unknown actual length) for Read access, using a delimiter table. ?VRNBR--If this flag is set to 1, the ring in offset ?VRING will be interpreted as a ring number (value 4 - 7). If this flag is set to 0, ?VRING will be interpreted as a ring field in bits 1S1-1S3 (value 0;4 to 0;7).
?VRES	Reserved. (Set to 0.)
?VPOINTER (double word)	Starting address of the area to be validated. This will be interpreted as a word address for ?RVWPL and ?WVWPL, and as a byte address for the other functions.
?VRING (double word)	Ring to be used for validation purposes. See ?VRNBR flag above for interpretation of this offset.
?VLENGTH (double word)	Length of the area to be validated. This will be interpreted as a word length for ?RVWPL and ?WVWPL, and as a byte length for the other functions.
?VDELIM (double word)	Word address of a 16-bit delimiter table to be used in validating areas of unknown actual length by the validation function ?RVBPX. You may also specify (-1) to designate the default delimiters (NULL, NEWLINE, and ERASEPAGE -- ASCII 0, 12, or 14). This parameter is ignored by the other validation functions.
?VERROR (double word)	Returns zero if the area was valid for the type of access specified. If this value is not zero, the area was invalid (see "Other Notes", next page).
?VRESD (double word)	Reserved for expansion. (Set to 0.)

The packet length is defined by the parameter ?VPLTH.

Error Conditions

The exception return for this call will be taken only if the call is invalid for one of the following reasons:

- It was called by a 16-bit process in Ring 7. ERICM — "Invalid System Command" — will indicate this error.
- The packet address in AC2 is invalid. ERVWP will be returned.
- The function code in ?VFUNC of the packet is unknown. ERUFR will be returned in this case.
- For function ?RVBPX only, if the contents of ?VDELIM is not (-1) or a valid address, error code ERVWP will be returned.

Note that the normal return indicates that the call was made correctly. However, it does NOT indicate the validity of the area. You must check the error code returned by ?VERROR, which will be zero if the type of access specified is possible. If ?VERROR is nonzero, it will be one of the following codes:

ERVWP This code is returned for functions ?RVWPL or ?WVWPL if the specified area is invalid for the indicated type of access.

ERVBP This code is returned for functions ?RVBPL, ?WVBPL, or ?RVBPX if the specified area is invalid for the indicated access type.

ERPRE will be reported for the following error conditions:

- The ring specified by ?VRING is invalid. See flag ?VRNBR for details of the format of the ring.
- The area length specified by ?VLEN is invalid. The area length must be a positive number which does not exceed the number of words or bytes in one ring.

Notes

- An area will be considered invalid if it crosses the unshared/shared boundary.

Examples of Use

Suppose an inner ring server for a database system gets a request to return some information from a database. The database caller, a Ring 7 application in this example, passes a byte address and byte length to specify the receiving area.

The database server may attempt to write the information into the customer's space without validation; however, it risks causing a USER TRAP from its ring. By calling ?VALIDATE with Ring 7 in ?VRING, and the user's byte address and length in ?VPTR and ?VLEN, the database server can simply return an error code for the request.

Suppose a Ring 7 caller provides a Ring 4 address to a Ring 4 server. ?VALIDATE should be set up to indicate that the caller's ring is 7; this "ring maximization" checking will return a validation error to protect the server.

Another example: Consider the situation where a string parameter, such as a filename, is an argument to some service. It is reasonable for the user to put the filename at the top of his unshared area, such that the string would be considered invalid if the entire length of a filename was used with function ?RVBPL. The user may have a null-terminated string "FOO<0>" which fits in this area. By using the function ?RVBPX and indicating the default delimiters table, the server can easily check the validity of this situation.

?VCUST

Verifies a customer in Ring 7.

?VCUST
error return
normal return

Input

AC0	PID of the process to verify
AC1	Reserved (Set to 0.)
AC2	Bits 0 through 28 are reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection does not exist
ERNAS	Process is not a server
ERRNI	Invalid ring number

Why Use It?

?VCUST gives you access to the system's connection table to find out whether a segment image of a process is a customer of Ring 7. ?VCUST is also a useful way to find out if a customer process has broken the connection from its end.

What It Does

?VCUST is a server-only system call. ?VCUST tests to see whether a segment image of the Ring 7 process that you specify in AC0 is a customer of the calling segment image.

If the target process in Ring 7 is the caller's customer and there is an unbroken connection between the two, control takes the normal return. If the target process in Ring 7 is not the caller's customer, ?VCUST fails on error code ERCDE. If the target process in Ring 7 is a customer, but the connection has been broken (for example, the customer issued a ?DCON system call), ?VCUST fails on error code ERCBK.

?VRCUST Verifies a customer in a specified ring.

?VRCUST
error return
normal return

Input

AC0	PID of the process to verify
AC1	Reserved (Set to 0.)
AC2	Bits 0 through 28 are reserved (Set to 0.) Bits 29 through 31 contain the ring number of the target in the specified process

Output

AC0	Unchanged
AC1	Undefined
AC2	Unchanged

Error Codes in AC0

ERCBK	Connection broken
ERCDE	Connection does not exist
ERNAS	Process is not a server
ERRNI	Invalid ring number

Why Use It?

?VRCUST gives you access to the system's connection table to find out whether a segment image of a process is a customer of the calling process ring. ?VRCUST is also a useful way to find out if a customer process has broken the connection from its end.

What It Does

?VRCUST is a server–only system call. ?VRCUST tests to see whether a segment image of the process that you specify in AC0 is a customer of the calling segment image.

If the target process ring is the caller's customer and there is an unbroken connection between the two, control takes the normal return. If the target process ring is not the caller's customer, ?VRCUST fails on error code ERCDE. If the target process ring is a customer, but the connection has been broken (for example, the customer issued a ?DCON system call), ?VRCUST fails on error code ERCBK.

?WALKBACK

Returns information about previous frames in the stack. (16–Bit Processes Only)

?WALKBACK
normal return

Input

AC0	Reserved (Set to 0.)
AC1	Reserved (Set to 0.)
AC2	One of the following: • 0 (first ?WALKBACK) • Unchanged (subsequent ?WALKBACKs)
AC3	Frame pointer with carry bit preserved (subsequent ?WALKBACKs only)

Output

AC0	One of the following: • Normal return for this environment if Bit 0 = 0 in AC1 • Procedure descriptor entry for this environment if Bit 0 = 1 in AC1
AC1	Bit 0 is a flag bit: • If Bit 0 = 0, resource call was not used to call this resource • If Bit 0 = 1, resource call was used to call this resource
AC2	One of the following: • Frame pointer for this stack level • 0 (which indicates that no stack frames are left)
AC3	Current frame pointer

Error Codes in AC0

No error codes are currently defined.

Licensed Material - Property of Data General Corporation

Why Use It?

?WALKBACK gives you a picture of the previous stack environment; that is, the contents of the stack after a previous ?KCALL, ?RCALL, or ?RCHAIN sequence. You can issue ?WALKBACK repeatedly, to obtain a complete picture of the stack, or until you reach the desired stack environment.

What It Does

?WALKBACK traces through the user stack and returns the stack frame pointer, procedure entry descriptor, and return address of the previous resource (the previous stack environment).

The first ?WALKBACK you issue retrieves the most recent stack environment. Before you issue the first ?WALKBACK, set AC2 to 0. On output, AC2 contains the frame pointer for the previous stack environment. Because the operating system uses the contents of the accumulators as input to subsequent ?WALKBACKs, do not alter the accumulators between ?WALKBACKs.

Notes

- See the descriptions of ?KCALL, ?RCALL, and ?RCHAIN in this chapter.

?WDELAY — Suspends a task for a specified time. (32–Bit Processes Only)

?WDELAY
error return
normal return

Input

AC0	Delay interval in milliseconds (a double–precision integer)
AC1	Reserved (Set to 0.)
AC2	Reserved (Set to 0.)

Output

AC0	Modified (used by the OS to maintain the delay)
AC1	Undefined
AC2	Undefined

Error Codes in AC0

No error codes are currently defined.

Why Use It?

Usually, you use ?WDELAY to synchronize tasks or to temporarily suspend a task until some asynchronous event has completed.

What It Does

?WDELAY suspends the calling task until the interval that you specify in AC0 has elapsed. If the delay interval is not a multiple of the real–time clock period, the operating system rounds it to the next higher multiple. For example, if the clock frequency is 10 hertz (one period is 100 milliseconds) and the delay interval that you specify is 120 milliseconds, the actual delay is 200 milliseconds. To obtain the current real–time clock frequency, issue ?GHRZ.

Licensed Material - Property of Data General Corporation

?WHIST

Starts a histogram. (32–Bit Processes Only)

?WHIST
error return
normal return

Input

AC0	One of the following: • Byte pointer to the name of the target process • PID of the target process • −1 to start a histogram for the calling process • 0 to start a histogram for all active processes
AC1	One of the following: • −1 if AC0 contains a byte pointer • Any other value, if AC0 contains −1, 0, or a PID
AC2	Address of the histogram packet

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERHIS	Error on histogram initialization or deletion
ERMPR	System call parameter address error
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERPRV	Caller not privileged for this action (The caller is not a resident process.)
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?WHIST allows you to monitor a range of addresses in the calling process or in another process for a specific interval. It also lets you compare the target's CPU time against the CPU time used by other processes, including the operating system. ?WHIST is a useful way to obtain a global view of CPU activity.

What It Does

?WHIST starts a histogram for a range of local addresses in a 32-bit process. (?IHIST is the analogous system call for 16-bit processes.) The caller must be a resident process and cannot activate more than one histogram at a time.

Before you issue ?WHIST, load AC0 with the PID of the target process or with a byte pointer to its process name. If you specify -1 in AC0, the operating system starts a histogram for the calling process. If you specify 0, the operating system compiles histogram statistics for all active processes. Note that your input to AC1 varies, depending on your input to AC0.

You must also set up a histogram packet of ?HWLTH words in your address space and reserve a receive buffer for the histogram statistics. Load the packet address into AC2 before you issue ?WHIST. Figure 2-214 shows the structure of the ?WHIST packet.

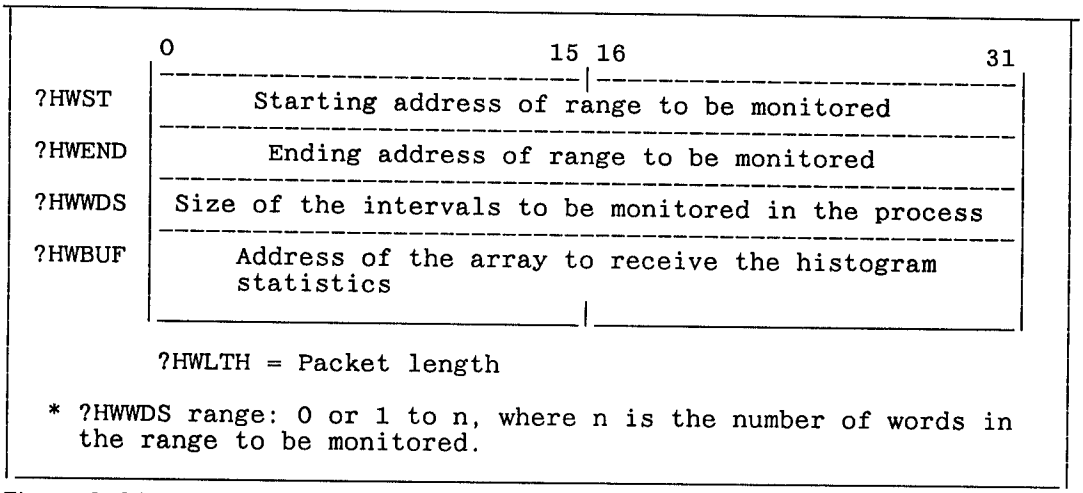


Figure 2-214. Structure of ?WHIST Packet

Set ?HWWDS to 0 for a simple histogram, which records how often the target process gained CPU control. The operating system ignores the contents of ?HWST and ?HWEND in a simple histogram.

The operating system returns the histogram statistics in array format to your receive buffer. The array consists of a fixed-length header, followed by double-precision elements that correspond to each interval you specify. Table 2-189 shows the array structure.

Table 2-189. Histogram Array Structure

Array Offset		Contents
?HTTH	?HTTL	Total number of real-time clock pulses (ticks) counted in this histogram.
?HPRH	?HPRL	Total number of ticks when 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		

Sample Packet

The following sample packet for a simple histogram shows how often the target process gained CPU control:

```
PKT:    .BLK    ?HWLTH          ;Allocate enough space for packet.
                                ;Packet length is ?HWLTH.
        .LOC    PKT+?HWST       ;Starting address of range to be
        .DWORD  0               ;monitored (ignored in simple
                                ;histogram).

        .LOC    PKT+?HWEND      ;Ending address of range to be
        .DWORD  0               ;monitored (ignored in simple
                                ;histogram).

        .LOC    PKT+?HWWDS      ;Size of interval to be monitored.
        .DWORD  0               ;Records how often process gained CPU
                                ;control.

        .LOC    PKT+?HWBUF      ;Address of array to receive
        .DWORD  0               ;histogram statistics (ignored in
                                ;simple histogram).

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

?WINDOW

Manipulates windows.

?WINDOW *[packet address]*
error return
normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?WINDOW packet

Error Codes Returned in AC0

ERBTS	Buffer too small
ERIVS	Invalid view or scan port
ERIWO	Invalid windowing operation
ERIWP	Invalid windowing parameter
ERIWR	Invalid window reference
ERPKT	Invalid packet ID
ERRVN	A reserved value was not set to 0
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid address passed as a system call argument
ERWMD	Window marked for deletion
ERWNE	The window you specified does not exist
ERWNN	Maximum number of windows exceeded

Error codes from the file system

Licensed Material - Property of Data General Corporation

Why Use It?

?WINDOW lets you create and manipulate windows on a physical terminal screen. You can also use ?WINDOW to control keyboard input to windows and to get information about windows.

?WINDOW is useful only if your program will run in an AOS/VS or AOS/DVS windowing environment. The call applies to a single window (or window group) at a time. For example, to find out the status of every window belonging to your program, you must issue the ?WINDOW call repeatedly, specifying a different window each time.

?WINDOW is a multifunctional system call. To perform a particular function, you set the offset ?WIN_PKT.FUNC (in the main ?WINDOW packet) to the function code you want. Table 2-190 lists the valid function codes and the functions they perform.

NOTE: Never use the system control processor in a window (unless you are debugging your own operating system).

Table 2-190. ?WINDOW Function Codes

Function Code	What It Lets You Do
?WIN_CREATE_WINDOW	Create a window on the physical terminal screen. (See the section "Creating a New Window.")
?WIN_DELETE_WINDOW	Delete an existing window. (See the section "Deleting a Window.")
?WIN_DEFINE_PORTS	Move or resize the view port or scan port of a window. This effectively lets you move the window on the screen, scroll data in the window, or change the size of the window. (See the section "Changing the View and Scan Ports.")
?WIN_UPFRONT_WINDOW	Make a window the highest priority in its group. (See the section "Controlling Window Priority.")
?WIN_OUTBACK_WINDOW	Make a window the lowest priority in its group. (See the section "Controlling Window Priority.")
?WIN_UPFRONT_GROUP	Make a group of windows the highest priority among groups on the physical terminal. (See the section "Controlling Window Priority.")
?WIN_OUTBACK_GROUP	Make a group of windows the lowest priority among groups on the physical terminal. (See the section "Controlling Window Priority.")
?WIN_HIDE_WINDOW	Make a window invisible to the user. (See the section "Controlling Window Visibility.")
?WIN_UNHIDE_WINDOW	Make a hidden window visible again. (See the section "Controlling Window Visibility.")
?WIN_HIDE_GROUP	Make a group of windows invisible to the user. (See the section "Controlling Window Visibility.")
?WIN_UNHIDE_GROUP	Make a hidden group of windows visible again. (See the section "Controlling Window Visibility.")
?WIN_SUSPEND_WINDOW	Suspend output to a window. (See the section "Controlling Output to a Window or Group.")
?WIN_UNSUSPEND_WINDOW	Resume output to a suspended window. (See the section "Controlling Output to a Window or Group.")
?WIN_SUSPEND_GROUP	Suspend output to a group of windows. (See the section "Controlling Output to a Window or Group.")
?WIN_UNSUSPEND_GROUP	Resume output to a suspended group of windows. (See "Controlling Output to a Window or Group.")

(Continues)

Table 2-190. ?WINDOW Function Codes

Function Code	What It Lets You Do
?WIN_ENABLE_KEYBOARD	Allow keyboard input to a window's input buffer. (See the section "Controlling Keyboard Input.")
?WIN_DISABLE_KEYBOARD	Prevent keyboard input from going to a window. (See the section "Controlling Keyboard Input.")
?WIN_PERMANENCE_ON	Set PERMANENCE on for a window. (See the section "Setting a Window's Permanence.")
?WIN_PERMANENCE_OFF	Set PERMANENCE off for a window. (See the section "Setting a Window's Permanence.")
?WIN_SET_USER_INTERFACE	Determine the appearance of the window border; specify whether or not the user can move, resize or scroll the window. (See the section "Setting the User Interface.")
?WIN_GET_USER_INTERFACE	Find out the current user interface settings for a window. (See the section "Getting the Current User Interface Settings.")
?WIN_SET_TITLE	Set the title of a window. (See the section "Setting a Window's Title.")
?WIN_GET_TITLE	Get the title of a window. (See the section "Getting a Window's Title.")
?WIN_GET_WINDOW_NAME	Get the pathname of a window. (See the section "Getting a Window's Pathname.")
?WIN_WINDOW_STATUS	Get the current status of a window. (See the section "Getting a Window's Status.")
?WIN_DEVICE_STATUS	Get the current status of a physical device. (See the section "Getting the Status of a Physical Device.")
?WIN_GET_WINDOW_ID	Get the window ID of a window. (See the section "Getting Window IDs.")
?WIN_RETURN_ GROUP_WINDOWS	Get the window IDs of all windows in a particular group. (See the section "Getting Window IDs.")
?WIN_RETURN_ DEVICE_WINDOWS	Get the window IDs of all windows on the physical terminal. (See the section "Getting Window IDs.")
?WIN_WRITE_PALETTE	Define colors in the palette associated with a window. (See the section "Writing to a Window's Palette.")
?WIN_READ_PALETTE	Find out what colors are stored in the palette associated with a window. (See the section "Reading a Window's Palette.")

(Concluded)

The Main ?WINDOW Packet

When you issue a ?WINDOW call, you usually set up both a main packet (common to all functions of the call) and a subpacket (unique for each function). Most functions require subpackets. The structure for the main packet appears in Figure 2-215; Table 2-191 details its contents. We cover the subpackets later, when we discuss each function.

The ?WINDOW call usually applies to one window at a time, or to a particular window group. In the main packet, you specify which window or group you want. You can specify a window in any of three ways:

- Channel number — When you open an I/O channel to a window (using ?OPEN), the operating system returns a channel number.
- Window pathname — When you create a window, you give it a window name. A window's pathname takes the form @PMAPn:windowname, where @PMAPn is the name of the pixel-mapped console that contains the window. The window pathname must end in a null character, <0>; it can be a maximum of ?MXPL characters long (including the null terminator).
- Window ID number — When you create a window, the operating system returns the window ID number in the main packet.

To specify a group of windows, simply specify a window within that group. Set the offset ?WIN_PKT.FLAGS to indicate which method you are using to specify the window or group. You can use only one method at a time. Fill in the appropriate field in the packet (channel number, pathname, or ID number), and set the other two fields to zero.

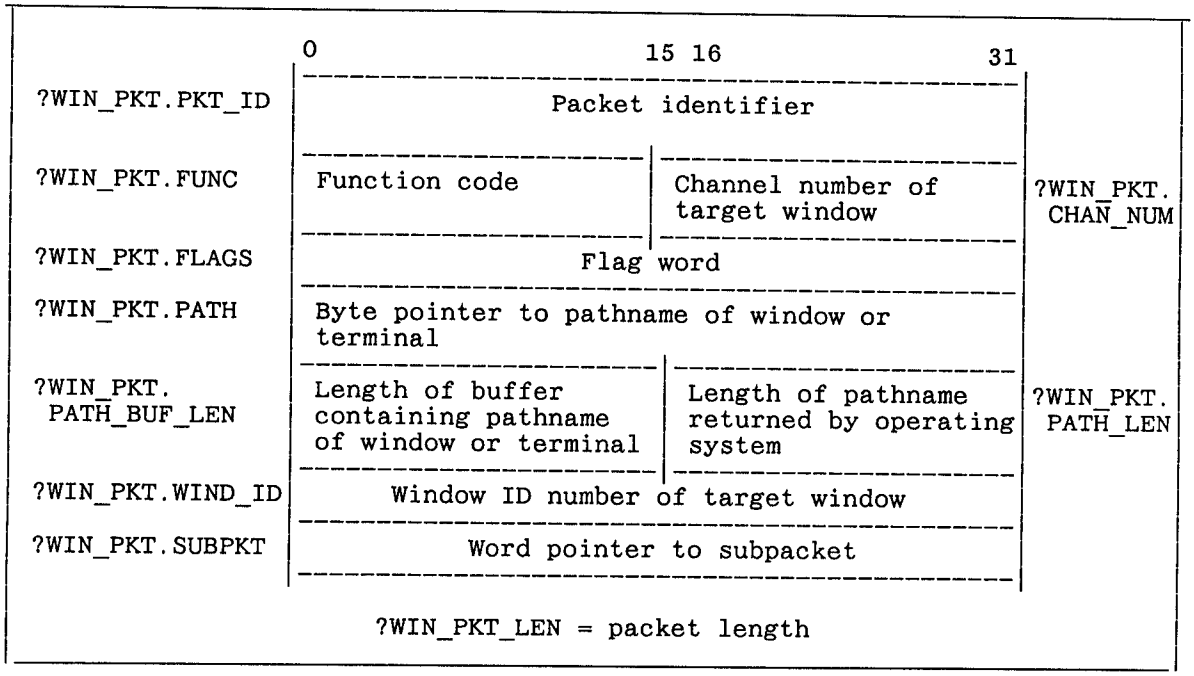


Figure 2-215. Structure of the ?WINDOW Main Packet

Table 2-191. Contents of the ?WINDOW Main Packet

Offset	Contents
?WIN_PKT.PKT_ID (double word)	Packet identifier; set to ?WIN_PKT_PKTID.
?WIN_PKT.FUNC	Code for the function you want. You must always supply a function code. (Function codes are listed in Table 2-190.)
?WIN_PKT.CHAN_NUM	Channel number of target window. When specifying the window by its channel number, you must also set flag ?WIN_PKT.FLAGS.IN_CHAN in flag word ?WIN_PKT.FLAGS. Set this offset to 0 when specifying the target window by its pathname or window ID.
?WIN_PKT.FLAGS (double word)	Flag word. ?WIN_PKT.FLAGS.IN_PATH -- Set this flag when specifying the target window by its pathname. ?WIN_PKT.FLAGS.IN_WIND_ID -- Set this flag when specifying target window by its window ID. ?WIN_PKT.FLAGS.IN_CHAN -- Set this flag when specifying target window by its channel number
?WIN_PKT.PATH (double word)	Byte pointer to pathname of target window. When specifying the window by its pathname, you must also set flag ?WIN_PKT.FLAGS.IN_PATH in flag word ?WIN_PKT.FLAGS. Set this offset to 0 when specifying the target window by its channel number or window ID.
?WIN_PKT.PATH_BUF_LEN	Length of the buffer containing the window pathname. The buffer length must include the null terminator; the maximum buffer length is ?MXPL.
?WIN_PKT.PATH_LEN	Unused. (Set to 0.)
?WIN_PKT.WIND_ID (double word)	Window ID number of target window. When specifying the window by its window ID, you must also set flag ?WIN_PKT.FLAGS.IN_WIND_ID in flag word ?WIN_PKT.FLAGS. Set this offset to 0 when specifying the target window by its pathname or channel number.
?WIN_PKT.SUBPKT	Word pointer to the subpacket for the function. If the function does not require a subpacket, set this offset to 0.

Creating a New Window

To create a new window on a physical terminal, issue the ?WINDOW call, function code ?WIN_CREATE_WINDOW. You can create a new window only if you have access to the physical terminal, or to an existing window on that terminal.

If you have access to the physical terminal, you can specify the terminal's device pathname in the main packet. The new window will then become the first window in a new window group. (If you specify a device pathname in the main packet, you must also set the flag ?WIN_CRE.FLAGS.CREATE_SON in the subpacket flag word ?WIN_CRE.FLAGS.)

If you have access to an existing window on the terminal, you can specify that window's pathname, channel number, or window ID in the main packet. The new window can either begin a new group or join the existing window's group.

In the ?WIN_CREATE_WINDOW subpacket, you must indicate whether you specified a window or a physical terminal in the main packet, and whether or not the new window will begin a new group. You can also specify some of the new window's attributes in the subpacket. Figure 2-216 shows the subpacket structure; Table 2-192 details its contents.

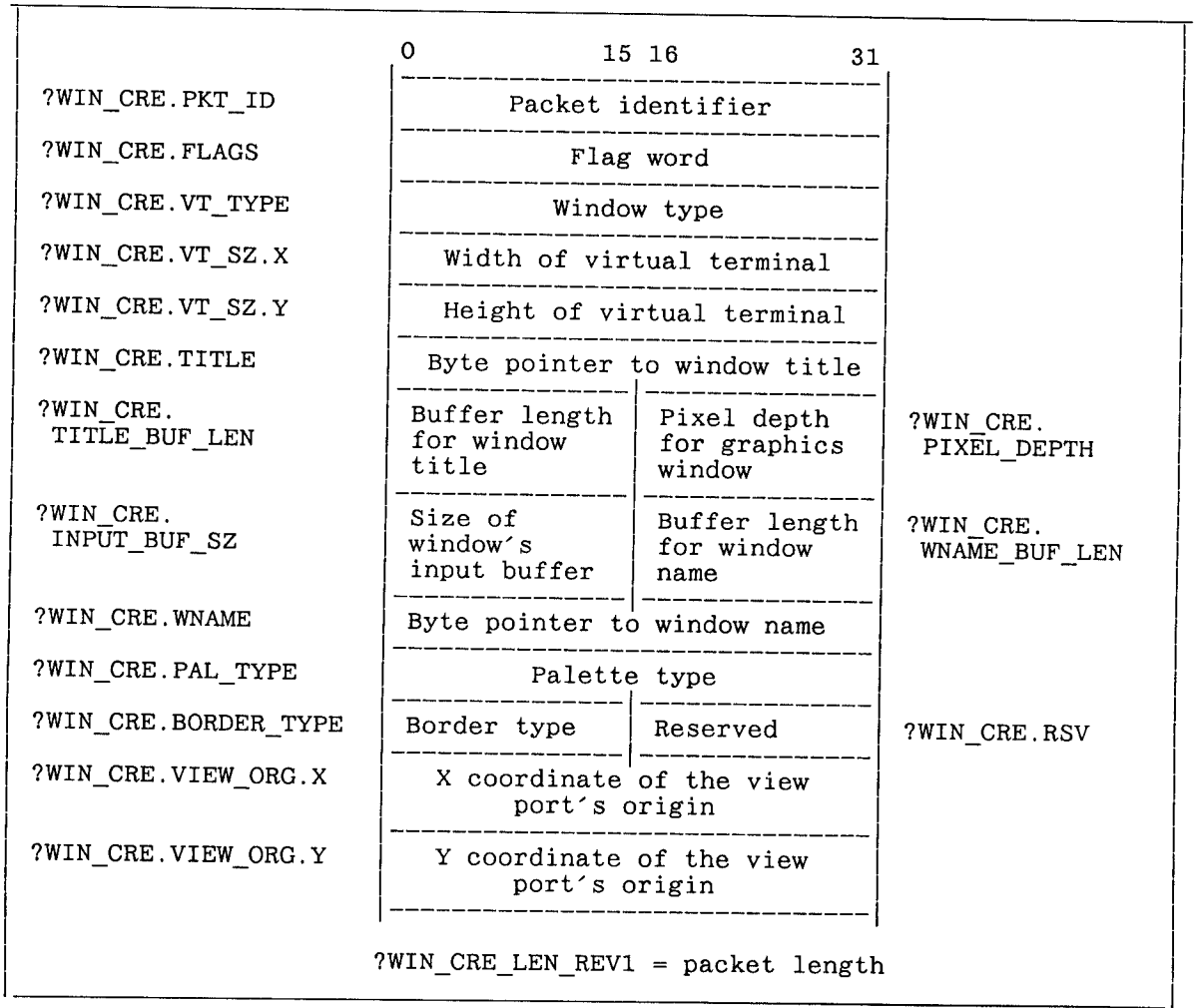


Figure 2-216. Structure of the ?WIN_CREATE_WINDOW Subpacket

Table 2-192. Contents of the ?WIN_CREATE_WINDOW Subpacket

Offset	Contents
?WIN_CRE.PKT_ID (double word)	Packet identifier; set to ?WIN_CRE_PKTID_REV1.
?WIN_CRE.FLAGS (double word)	Flag word. ?WIN_CRE.FLAGS.NEW_GRP -- Set this flag if the new window will be the first window in a new group. You must set this flag if you specify a physical terminal's pathname in the main packet. ?WIN_CRE.FLAGS.VT_SZ_X -- Set this flag if you are specifying the width of the virtual terminal; set offset ?WIN_CRE.VT_SZ_X to the value you want. If you do not set this flag, the operating system uses the default width. ?WIN_CRE.FLAGS.VT_SZ_Y -- Set this flag if you are specifying the width of the virtual terminal; set offset ?WIN_CRE.VT_SZ_Y to the value you want. If you do not set this flag, the operating system uses the default height. ?WIN_CRE.FLAGS.PIXEL_DEPTH -- Set this flag if you are specifying the number of bits per pixel for a graphics window; set offset ?WIN_CRE.PIXEL_DEPTH to the value you want. If you do not set this flag, the operating system uses the default pixel depth. ?WIN_CRE.FLAGS.CREATE_SON -- You must set this flag if you specify a physical terminal's pathname in the main packet (the new window will be a "son window" of the terminal). If you set this flag, you must also set the flag ?WIN_CRE.FLAGS.NEW_GRP. ?WIN_CRE.FLAGS.VIEW_ORG.X -- You must set this flag if you are specifying an X coordinate for the view port's origin; set offset ?WIN_CRE.VIEW_ORG.X to the value you want. If you do not set this flag, the operating system uses a default value. ?WIN_CRE.FLAGS.VIEW_ORG.Y -- You must set this flag if you are specifying a Y coordinate for the view port's origin; set offset ?WIN_CRE.VIEW_ORG.Y to the value you want. If you do not set this flag, the operating system uses a default value. ?WIN_CRE.FLAGS.UNHIDE_WINDOW -- Set this flag to make the window visible immediately after creation. If you do not set this flag, the window remains hidden until you use the ?WINDOW function ?WIN_UNHIDE_WINDOW to make it visible.

(Continues)

Table 2-192. Contents of the ?WIN_CREATE_WINDOW Subpacket

Offset	Contents
?WIN_CRE.FLAGS (double word) (continues)	?WIN_CRE.FLAGS.INPUT_BUF_SZ -- Set this flag if you are specifying the size of the window's input buffer; set offset ?WIN_CRE.INPUT_BUF_SZ to the value you want. If you do not set this flag, the operating system uses the size of the input buffer associated with the physical terminal.
?WIN_CRE.VT_TYPE (double word)	Type of virtual terminal the new window will have. Select from the following values. ?WIN_VT_TYPE_DEFAULT — The operating system determines the virtual terminal type. ?WIN_VT_TYPE_D460 — (Character window with 2 bits per pixel.) The virtual terminal will be a software emulation of a D460 terminal and will support the blink attribute. ?WIN_VT_TYPE_MONO_D460 — (Character window with 1 bit per pixel.) The virtual terminal will be a software emulation of a D460 terminal but will not support the blink attribute. ?WIN_VT_TYPE_PIXMAP — (Graphics window.) The virtual terminal will be a pixel map.
?WIN_CRE.VT_SZ.X (double word)	Width of the new window's virtual terminal. For a character window (?WIN_VT_TYPE_D460), specify the width in characters. For a graphics window (?WIN_VT_TYPE_PIXMAP), specify the width in pixels. When specifying virtual terminal width, you must set the flag ?WIN_CRE.FLAGS.VT_SZ_X in flag word ?WIN_CRE.FLAGS.
?WIN_CRE.VT_SZ.Y (double word)	Height of the new window's virtual terminal. For a character window (?WIN_VT_TYPE_D460), specify the height in characters. For a graphics window (?WIN_VT_TYPE_PIXMAP), specify the height in pixels. When specifying virtual terminal height, you must set the flag ?WIN_CRE.FLAGS.VT_SZ_Y in flag word ?WIN_CRE.FLAGS.

(Continued)

Table 2-192. Contents of the ?WIN_CREATE_WINDOW Subpacket

Offset	Contents
?WIN_CRE.TITLE (double word)	Byte pointer to the buffer containing the new window's title. The window title is a character string up to 32 characters long; the last character must be a null. You must always provide a byte pointer. To create a window with no title, provide a byte pointer to a null title string.
?WIN_CRE. TITLE_BUF_LEN	The length of the buffer containing the window title. The maximum length for the window title is ?MXFN.
?WIN_CRE. PIXEL_DEPTH	The number of bits per pixel, for a graphics window.
?WIN_CRE. INPUT_BUF_SZ	The length of the input buffer associated with the new window's virtual terminal. The buffer length must be at least 32. When specifying input buffer length, you must also set the flag ?WIN_CRE.FLAGS.INPUT_BUF_SZ in flag word ?WIN_CRE.FLAGS.
?WIN_CRE. WNAME_BUF_LEN	The length of the buffer containing the window name. The maximum length for the window name is ?MXFN.
?WIN_CRE.WNAME (double word)	Byte pointer to the buffer containing the new window's window name. The window name is a character string up to ?MXFN characters long. You must always provide a window name, since it forms part of the window's pathname.
?WIN_CRE.PAL_TYPE (double word)	Type of palette the new window will have. Select from the following values. ?WIN_PALETTE_TYPE_DEFAULT -- The system determines the palette type for this window. ?WIN_PALETTE_TYPE_TRUE -- True-color palette. Supported only on 24-bit per pixel terminals. The window cannot write to this palette. ?WIN_PALETTE_TYPE_USER -- User-definable palette. Has 2**(pixel depth) entries. For character windows, the pixel depth depends on the virtual terminal type. For graphics windows, you specify the pixel depth in offset ?WIN_CRE.PIXEL_DEPTH. All entries in the palette are initially set to "dark".

(Continued)

Table 2-192. Contents of the ?WIN_CREATE_WINDOW Subpacket

Offset	Contents
	?WIN_PALETTE_TYPE_SYSTEM -- System palette. The window shares the system palette, which is two entries long. The window cannot write to this palette. ?WIN_PALETTE_TYPE_D460 -- D460 palette. Has four entries, initially set to "dark," "bright," "dim," and "blink." ?WIN_PALETTE_TYPE_MONO_D460 -- Monochrome D460 palette. Has two entries, initially set to "dark" and "bright." ?WIN_PALETTE_TYPE_SHARED -- Shared palette. The window shares the palette of the existing window specified in the main ?WINDOW packet.
?WIN_CRE. BORDER_TYPE	Type of border the new window will have. Select from the following values. ?WIN_BORDER_TYPE_DEFAULT — System determined border type. ?WIN_BORDER_TYPE_NONE — No border. ?WIN_BORDER_TYPE_LINE — Simple line border. ?WIN_BORDER_TYPE_TITLE — Line border with space at the top to display a window title ?WIN_BORDER_TYPE_INTELLIGENT — Intelligent border.
?WIN_CRE. VIEW_ORG.X (double word)	The initial X coordinate of the view port's origin (in pixels relative to the origin of the physical screen). To use the system default value, set to 0.
?WIN_CRE. VIEW_ORG.Y (double word)	The initial Y coordinate of the view port's origin (in pixels relative to the origin of the physical screen). To use the system default value, set to 0.

(Concluded)

Deleting a Window

To delete an existing window, issue ?WINDOW with function code ?WIN_DELETE_WINDOW. If the window is in use (that is, if a channel is open to it, or it is still assigned to a process) the operating system will not delete it until the last process deassigns it. In the main packet, provide a pathname, channel number or window ID for the window you want to delete. There is no subpacket for ?WIN_DELETE_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

Changing the View and Scan Ports

Every window has a view port (the portion of the physical screen within the window border) and a scan port (the portion of the virtual screen that's within the view port). The view port is always the same size as the scan port.

You can alter the window's size and position by changing the size and location of its ports. To do so, issue the ?WINDOW call with function code ?WIN_DEFINE_PORTS; in the main packet, provide a pathname, channel number or window ID for the window you want to change.

- To move the window on the physical screen, alter the origin of its view port.
- To scroll data in the window (move the scan port on the virtual screen), alter the origin of its scan port.
- To change the size of the window, alter the size of its scan port; the operating system adjusts the size of the view port to match.

You specify coordinate values in the ?WIN_DEFINE_PORTS subpacket. Figure 2-217 shows the subpacket structure; Table 2-193 details its contents.

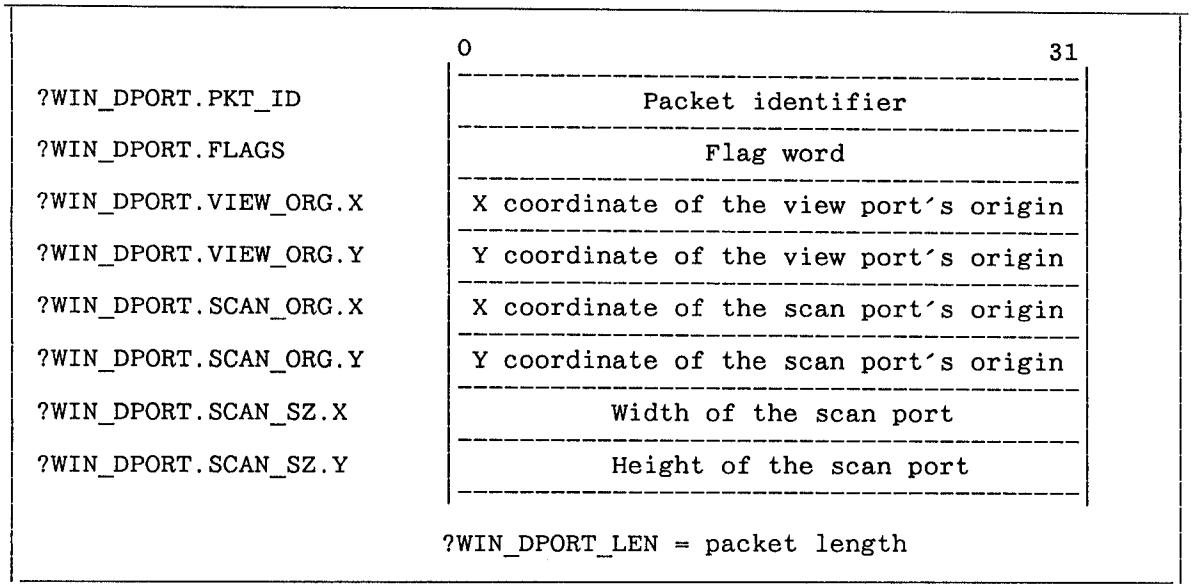


Figure 2-217. Structure of the ?WIN_DEFINE_PORTS Subpacket

For each view or scan port parameter you want to change, you must set the corresponding flag in the flag word ?WIN_DPORT.FLAGS. If you are not changing a particular parameter, set both the flag and the parameter offset to 0.

Table 2-193. Contents of the ?WIN_DEFINE_PORTS Subpacket

Offset	Contents
?WIN_DPORT.PKT_ID (double word)	Packet identifier; set to ?WIN_DPORT_PKTID.
?WIN_DPORT.FLAGS (double word)	Flag word. For each value you change, you must set the corresponding flag, as follows. ?WIN_DPORT.FLAGS.VIEW_ORG_X -- Set this flag if you are changing the X coordinate of view port's origin. ?WIN_DPORT.FLAGS.VIEW_ORG_Y -- Set this flag if you are changing the Y coordinate of view port's origin. ?WIN_DPORT.FLAGS.SCAN_ORG_X -- Set this flag if you are changing the X coordinate of scan port's origin. ?WIN_DPORT.FLAGS.SCAN_ORG_Y -- Set this flag if you are changing the Y coordinate of scan port's origin. ?WIN_DPORT.FLAGS.SCAN_SZ_X -- Set this flag if you are changing the width of the scan port. ?WIN_DPORT.FLAGS.SCAN_SZ_Y -- Set this flag if you are changing the height of the scan port.
?WIN_DPORT.VIEW_ORG.X (double word)	X coordinate of the view port's origin (in pixels relative to the origin of the physical screen). Set to 0 if you are not changing this value.
?WIN_DPORT.VIEW_ORG.Y (double word)	Y coordinate of the view port's origin (in pixels relative to the origin of the physical screen). Set to 0 if you are not changing this value.
?WIN_DPORT.SCAN_ORG.X (double word)	X coordinate of the scan port's origin (in pixels for a graphics window, characters for a character window); relative to the origin of the virtual screen. Set to 0 if you are not changing this value.
?WIN_DPORT.SCAN_ORG.Y (double word)	Y coordinate of the scan port's origin (in pixels for a graphics window, characters for a character window); relative to the origin of the virtual screen. Set to 0 if you are not changing this value.
?WIN_DPORT.SCAN_SZ.X (double word)	Width of the scan port (in pixels for a graphics window, characters for a character window). Set to 0 if you are not changing this value.
?WIN_DPORT.SCAN_SZ.Y (double word)	Height of the scan port (in pixels for a graphics window, characters for a character window). Set to 0 if you are not changing this value.

Licensed Material - Property of Data General Corporation

Controlling Window Priority

The operating system assigns a unique window priority to each window on the screen. The window priority determines the front-to-back order of the windows: where windows overlap, the window with the higher priority appears in front of any window(s) with lower priorities. Window groups are collections of windows that the operating system maintains at consecutive priorities. Each group can be considered to have a priority relative to other groups.

You use the ?WINDOW call to change the priority of a window within its group, or to change the priority of a window group relative to other groups on the screen.

To Bring a Window or Group to the Front

The ?WIN_UPFRONT_WINDOW function makes a window the highest priority among windows in its group; when visible, that window will then appear in front of other windows in its group. (This function does not alter the window's visibility.) In the main packet, provide a pathname, channel number or window ID for the window you want to bring to the front. There is no subpacket for ?WIN_UPFRONT_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

The ?WIN_UPFRONT_GROUP function makes a window group the highest priority among groups on the screen. When that group is visible, it will be the active group, and its windows will appear in front of windows belonging to other groups. (This function does not alter the group's visibility or the internal hierarchy of its windows.) In the main packet, provide a pathname, channel number or window ID for any window in the group you want to bring to the front. There is no subpacket for ?WIN_UPFRONT_GROUP; set offset ?WIN_PKT.SUBPKT to 0.

To Send a Window or Group to the Back

The ?WIN_OUTBACK_WINDOW function makes a window the lowest priority among windows in its group; when visible, that window will then appear behind other windows in its group. In the main packet, provide a pathname, channel number or window ID for the window you want to send to the back. There is no subpacket for ?WIN_OUTBACK_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

The ?WIN_OUTBACK_GROUP function makes a group of windows the lowest priority among groups on the screen; when that group is visible, its windows will then appear behind windows in other groups. If you issue this function for the active group, the operating system makes the visible group with the highest priority the new active group. In the main packet, provide a pathname, channel number or window ID for any window in the group you want to send to the back. There is no subpacket for ?WIN_OUTBACK_GROUP; set offset ?WIN_PKT.SUBPKT to 0.

Controlling Window Visibility

A window on the screen can be either visible or invisible to the user. The window's visibility does not affect its priority (its place within the hierarchy of windows in its group). Likewise, hiding a group of windows does not affect that group's priority relative to other groups, or its internal hierarchy of windows.

To Make a Window or Group Invisible

The ?WIN_HIDE_WINDOW function makes a window invisible to the user. Although a hidden window does not appear on the screen, it keeps its place in the hierarchy of windows within its group.

In the main packet, provide a pathname, channel number or window ID for the window you want to hide. There is no subpacket for ?WIN_HIDE_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

The ?WIN_HIDE_GROUP function makes a group of windows invisible to the user. This function does not affect the group's place within the hierarchy of groups on the screen, or the hierarchy of windows within the group. If you hide the active group, the visible group with the highest priority becomes the new active group.

In the main packet, provide a pathname, channel number or window ID for any window in the group you want to hide. There is no subpacket for ?WIN_HIDE_GROUP; set offset ?WIN_PKT.SUBPKT to 0.

To Make a Hidden Window or Group Visible

The ?WIN_UNHIDE_WINDOW function makes a hidden window visible again. In the main packet, provide a pathname, channel number or window ID for the window you want to make visible. There is no subpacket for ?WIN_UNHIDE_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

The ?WIN_UNHIDE_GROUP function makes a hidden group of windows visible to the user. (This function does not affect the group's priority relative to other groups, or the internal hierarchy of windows within that group. It also does not affect the visibility of individual windows: if a window in the group was previously hidden individually, it remains hidden even after you make its group visible.)

In the main packet, provide a pathname, channel number or window ID for any window in the group you want to make visible. There is no subpacket for ?WIN_UNHIDE_GROUP; set offset ?WIN_PKT.SUBPKT to 0.

Controlling Output

You can use ?WINDOW to suspend or resume output to a window's virtual screen. When you suspend a window, output to that window's virtual screen stops as if the user had pressed CTRL-S. You can suspend either a single window or a group of windows. When you resume output to a window or group, output to the virtual screen(s) resumes as if the user had pressed CTRL-Q.

To Suspend Output

To suspend output to a window, issue the ?WINDOW call with function code ?WIN_SUSPEND_WI NDOW. In the main packet, provide a pathname, channel number or window ID for the window you want to suspend. There is no subpacket for ?WIN_SUSPEND_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

To suspend output to an entire group of windows, issue ?WINDOW with function code ?WIN_SUSPEND_GROUP. In the main packet, provide a pathname, channel number or window ID for any window in the group you want to suspend. There is no subpacket for ?WIN_SUSPEND_GROUP; set offset ?WIN_PKT.SUBPKT to 0.

To Resume Output

To resume output to a suspended window, issue ?WINDOW with function code ?WIN_UNSUSPEND_WINDOW. In the main packet, provide a pathname, channel number or window ID for the window to which you want to resume output. There is no subpacket for ?WIN_UNSUSPEND_WINDOW; set offset ?WIN_PKT.SUBPKT to 0.

To resume output to a suspended group of windows, issue ?WINDOW with function code ?WIN_UNSUSPEND_GROUP. (If a window in the group was previously suspended individually, it remains suspended until you issue a ?WINDOW call, function code ?WIN_UNSUSPEND_WINDOW.) In the main packet, provide a pathname, channel number or window ID for any window in the group to which you want to resume output. There is no subpacket for ?WIN_UNSUSPEND_GROUP; set offset ?WIN_PKT.SUBPKT to 0.

Controlling Keyboard Input

?WINDOW lets you control whether or not keyboard input enters a window's input data stream. By default, keyboard input is enabled.

To prevent keyboard input from entering a window's input data stream, issue ?WINDOW, function code ?WIN_DISABLE_KEYBOARD. In the main packet, provide a pathname, channel number or window ID for the window you want to disable. There is no subpacket for ?WIN_DISABLE_KEYBOARD; set offset ?WIN_PKT.SUBPKT to 0.

To allow keyboard input to go to a window for which you previously disabled keyboard input, issue ?WINDOW with function code ?WIN_ENABLE_KEYBOARD. In the main packet, provide a pathname, channel number or window ID for the window you want to enable. There is no subpacket for ?WIN_ENABLE_KEYBOARD; set offset ?WIN_PKT.SUBPKT to 0.

Setting a Window's Permanence

Windows have a PERMANENCE attribute just as files do. When a window's PERMANENCE is on, you cannot delete that window.

Use function ?WIN_PERMANENCE_ON to turn PERMANENCE on for a window. In the main packet, provide a pathname, channel number or window ID for the window you want to protect. There is no subpacket for ?WIN_PERMANENCE_ON; set offset ?WIN_PKT.SUBPKT to 0.

Use function ?WIN_PERMANENCE_OFF to turn window PERMANENCE off. In the main packet, provide a pathname, channel number or window ID for the window you want. There is no subpacket for ?WIN_PERMANENCE_OFF; set offset ?WIN_PKT.SUBPKT to 0.

Setting the User Interface

You can control some aspects of a window's appearance using the ?WINDOW call with function code ?WIN_SET_USER_INTERFACE. This function lets you determine

- The type of window border (intelligent, title, line, or none).
- Whether or not the user can move the window.
- Whether or not the user can change the window's size.
- Whether or not the user can scroll data in the window.
- Whether or not to maintain elevators on an intelligent border.
- Whether or not the user can request help by selecting the help symbol on an intelligent border.
- Whether or not the user can make a window full sized and return it to its original size.
- Whether or not the user can move the borders off the screen.

NOTE: The user alters windows using DG/VIEW, a menu-driven windowing interface. If DG/VIEW is not available, the user will not be able to change the window, regardless of the settings in the ?WIN_SET_USER_INTERFACE subpacket.

Licensed Material - Property of Data General Corporation

In the main packet, specify the window ID, pathname or channel number of the window you want to change. You specify the changes you want in the subpacket. Figure 2-218 shows the structure of the subpacket; Tables 2-194 and 2-195 detail its contents.

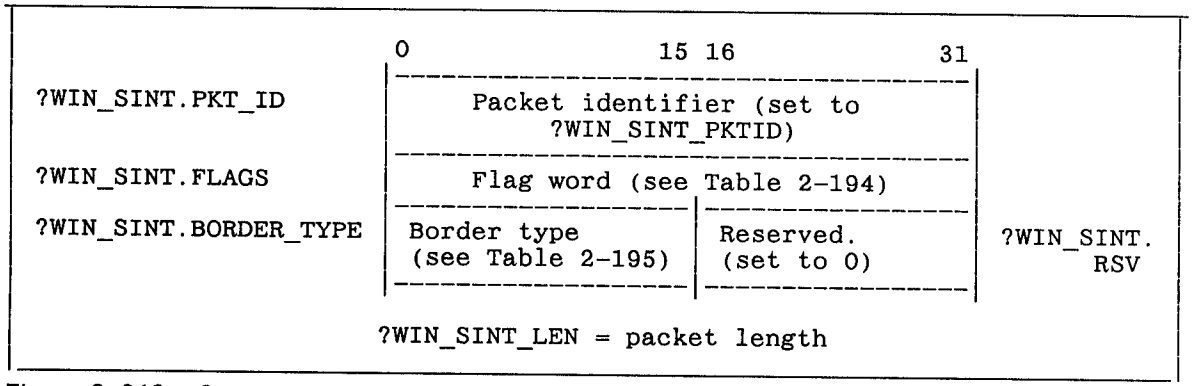


Figure 2-218. Structure of the ?WIN_SET_USER_INTERFACE Subpacket

Table 2-194. Flags in Flag Word ?WIN_SINT.FLAGS

Flag	Description
?WIN_SINT.FLAGS.MOVE_X	Set this flag to let the user move the window along the x-axis.
?WIN_SINT.FLAGS.MOVE_Y	Set this flag to let the user move the window along the y-axis.
?WIN_SINT.FLAGS.SZ_X	Set this flag to let the user change the width of the window.
?WIN_SINT.FLAGS.SZ_Y	Set this flag to let the user change the height of the window.
?WIN_SINT.FLAGS.SCROLL_X	Set this flag to let the user scroll data horizontally in the window.
?WIN_SINT.FLAGS.SCROLL_Y	Set this flag to let the user scroll data vertically in the window.
?WIN_SINT.FLAGS.ELEVATORS	Set this flag to maintain elevators on the intelligent border.
?WIN_SINT.FLAGS.FULL_REV	Set this flag to let the user expand the window to full size and revert it back to its original size.
?WIN_SINT.FLAGS.HELP	Set this flag to let the user select the help symbol on the intelligent border.
?WIN_SINT.FLAGS.WIRE_BRDRS	Set this flag to prevent the user from moving window borders off the screen.

Table 2-195. Border Types (Offset ?WIN_SINT.BORDER_TYPE)

Value	Description
?WIN_BORDER_TYPE_DEFAULT	System determined border type.
?WIN_BORDER_TYPE_NONE	No border.
?WIN_BORDER_TYPE_LINE	Simple line border.
?WIN_BORDER_TYPE_TITLE	Line border with space at the top to display a window title.
?WIN_BORDER_TYPE_INTELLIGENT	Intelligent border.

Getting the Current User Interface Settings

To find out a window's current user interface settings, issue the ?WINDOW call with function code ?WIN_GET_USER_INTERFACE. In the main packet, specify the window ID, pathname or channel number of the window whose user interface settings you want. The operating system returns the information in the subpacket. Figure 2-219 shows the structure of the subpacket; Tables 2-196 and 2-197 detail its contents.

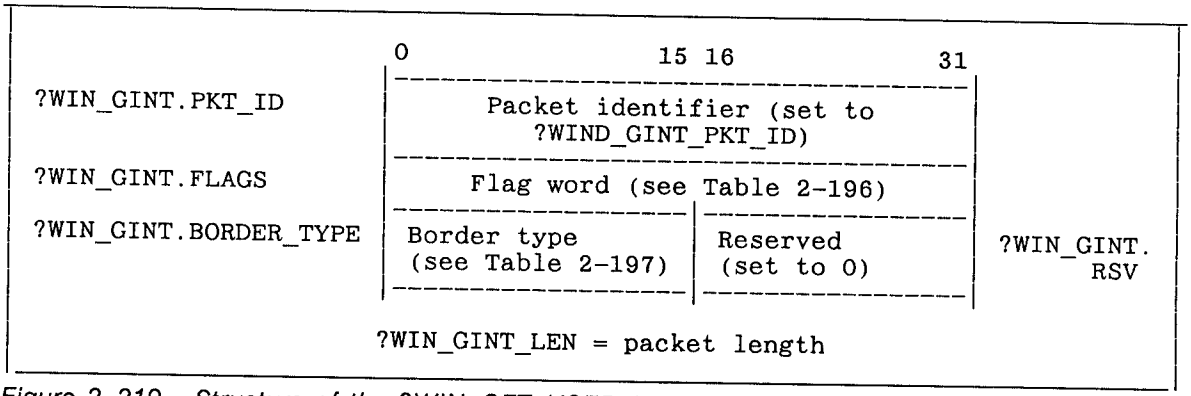


Figure 2-219. Structure of the ?WIN_GET_USER_INTERFACE Subpacket

NOTE: The user alters windows using DG/VIEW, a menu-driven windowing interface. If DG/VIEW is not available, the user will not be able to change the window, regardl ess of the settings returned in the ?WIN_GET_USER_INTERFACE subpacket.

Table 2-196. Flags in the ?WIN_GET_USER_INTERFACE Flag Word (Flag Word ?WIN_GINT.FLAGS)

Flag	Description
?WIN_GINT.FLAGS.MOVE_X	This flag is set if the user can move the window horizontally.
?WIN_GINT.FLAGS.MOVE_Y	This flag is set if the user can move the window vertically.
?WIN_GINT.FLAGS.SZ_X	This flag is set if the user can change the width of the window.
?WIN_GINT.FLAGS.SZ_Y	This flag is set if the user can change the height of the window.
?WIN_GINT.FLAGS.SCROLL_X	This flag is set if the user can scroll data horizontally in the window.
?WIN_GINT.FLAGS.SCROLL_Y	This flag is set if the user can scroll data vertically in the window.
?WIN_GINT.FLAGS.ELEVATORS	This flag is set if elevators are maintained on the intelligent border.
?WIN_GINT.FLAGS.FULL_REV	This flag is set if the user can expand the window to full size and revert it back to its original size.
?WIN_GINT.FLAGS.HELP	This flag is set if the user can select the help symbol on the intelligent border.
?WIN_GINT.FLAGS.WIRE_BRDRS	This flag is set if the user cannot move window borders off the screen.

Table 2-197. Border Types (Offset ?WIN_GINT.BORDER_TYPE)

Value	Description
?WIN_BORDER_TYPE_DEFAULT	System determined border type.
?WIN_BORDER_TYPE_NONE	No border.
?WIN_BORDER_TYPE_LINE	Simple line border.
?WIN_BORDER_TYPE_TITLE	Line border with space at the top to display a window title.
?WIN_BORDER_TYPE_INTELLIGENT	Intelligent border.

Setting a Window's Title

If a window has an intelligent or a title border, a window title can appear within the border. The window title only identifies the window to the user; you cannot refer to a window by its title when issuing a system call.

To set the window title, issue ?WINDOW with function code ?WIN_SET_TITLE; in the main packet, provide a pathname, channel number, or window ID for the window whose title you want to set. In the subpacket, specify the size and location of the buffer that contains the title string.

The title string always ends in a null, and can be up to ?MXFN characters long (including the null terminator). Every window has a title string; the title string for a window with no title is a null string.

Figure 2–220 shows the subpacket structure.

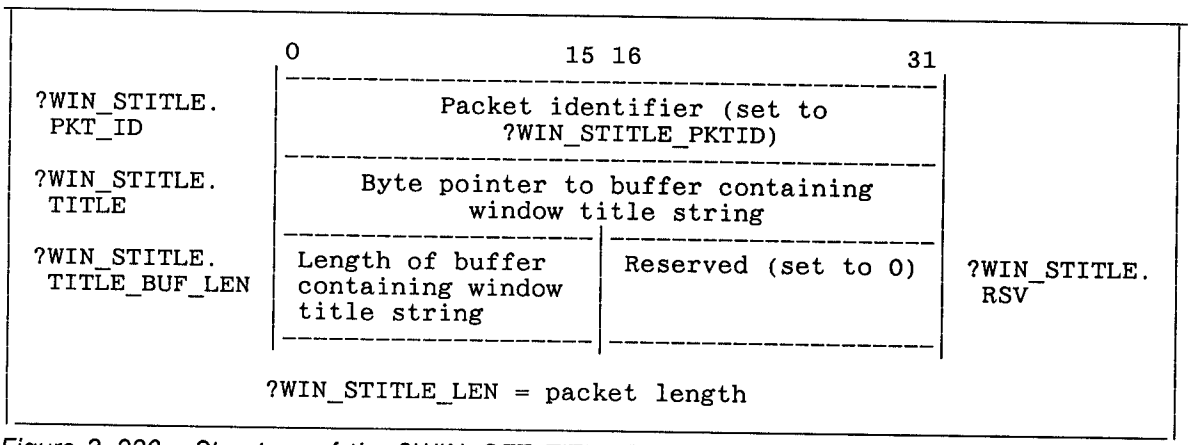


Figure 2–220. Structure of the ?WIN_GET_TITLE Subpacket

Getting a Window's Title

To find out the current title of a window, issue ?WINDOW with function code ?WIN_GET_TITLE; in the main packet, provide a pathname, channel number or window ID for the window whose title you want. In the subpacket, specify the size and location of the buffer that will contain the title string. (A window's title can be up to ?MXFN characters long, including the null terminator. Every window has a title string; the title string for a window with no title is a null string.)

The operating system returns a byte pointer to the window's title string. If the buffer you specified was too small to contain the string, the operating system takes the error return, and returns the length of the title string in offset ?WIN_GTITLE_TITLE_LEN.

Figure 2–221 shows the subpacket structure; Table 2–198 details its contents.

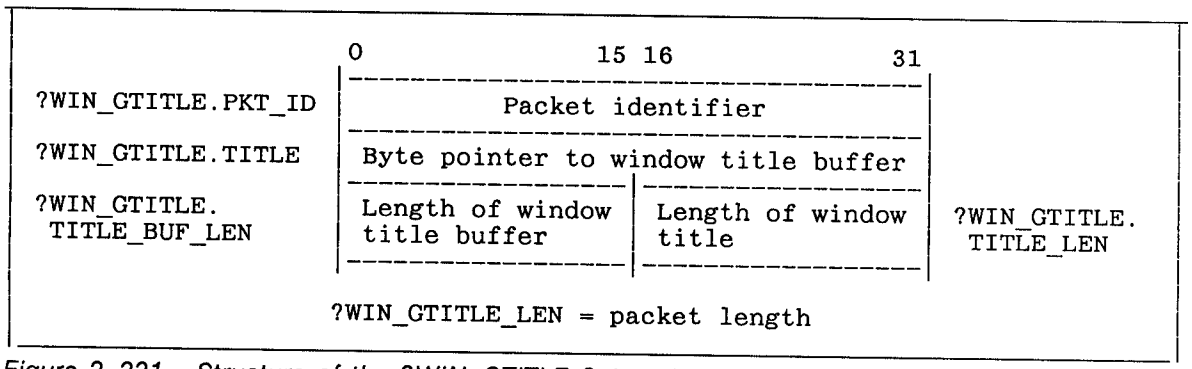


Figure 2-221. Structure of the ?WIN_GTITLE Subpacket

Table 2-198. Contents of the ?WIN_GTITLE Subpacket

Offset	Contents
?WIN_GTITLE.PKT_ID (double word)	Packet identifier; set to ?WIN_GTITLE_PKT_ID.
?WIN_GTITLE.TITLE (double word)	Provide a byte pointer to the buffer that will contain the title string returned by the operating system.
?WIN_GTITLE.TITLE_BUF_LEN	Supply the length of the buffer that will contain the title string. The maximum title length is ?MXFN.
?WIN_GTITLE.TITLE_LEN	Set to 0. If the title buffer is too short to hold the title string, the operating system returns the length of the string here (including the null terminator).

To find out a window's pathname, issue ?WINDOW with function code ?WIN_GET_WINDOW_NAME; in the main packet, specify the window ID or channel number of the window you want. There is no subpacket for the ?WIN_GET_WINDOW_NAME function; set ?WIN_PKT.SUBPKT to 0.

The operating system returns a byte pointer to the window's pathname in offset ?WIN_PKT.PATH. It returns the length of the pathname (including the null terminator) in ?WIN_PKT.PATH_LEN.

Getting a Window's Status

The ?WINDOW function ?WIN_WINDOW_STATUS returns the following information about a window:

- Whether or not the window (or its group) is hidden.
- Whether or not the window (or its group) is suspended .
- Whether or not the window's keyboard is enabled.
- Whether or not window PERMANENCE is on.
- The size and location of the window's view port.
- The size and location of the window's scan port.
- The size and location of the window on the screen.
- The window's pixel depth (for graphics windows).
- The size of the window's input buffer.
- The type of palette associated with the window.

In the main packet, specify the window ID, pathname or channel number of the window you want. The operating system returns information in the subpacket. Figure 2–222 shows the structure of the subpacket; Table 2–199 details its contents.

Licensed Material - Property of Data General Corporation

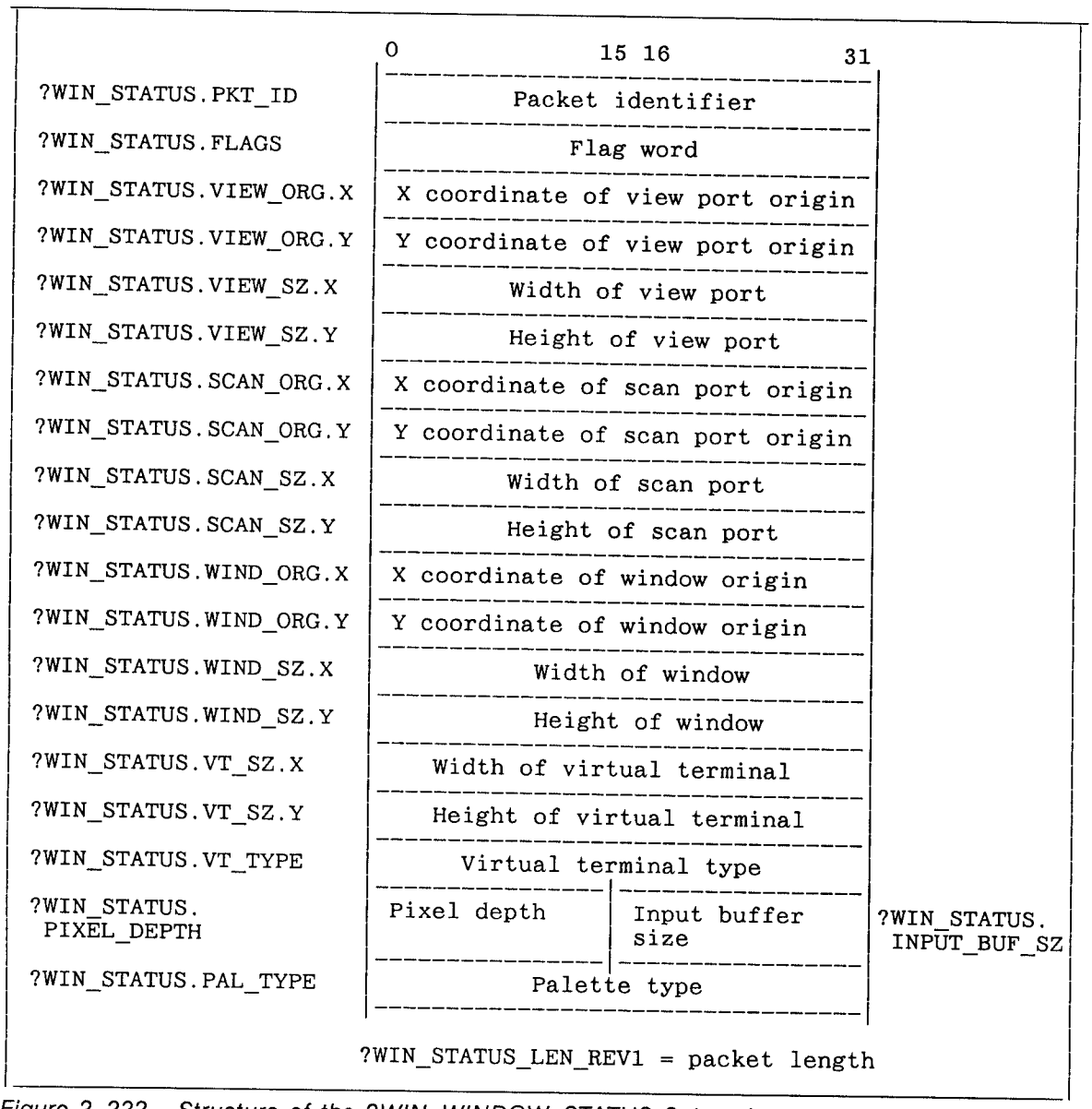


Figure 2-222. Structure of the ?WIN_WINDOW_STATUS Subpacket

Table 2-199. Contents of the ?WIN_WINDOW_STATUS Subpacket

Offset	Contents
?WIN_STATUS.PKT_ID (double word)	Packet identifier; set to ?WIN_STATUS_PKTID.
?WIN_STATUS.FLAGS (double word)	Flag word. Flag settings indicate the current status of window attributes, as follows: ?WIN_STATUS.FLAGS.GRP_HIDDEN – Window's group is invisible. ?WIN_STATUS.FLAGS.WIND_HIDDEN – Window is invisible. ?WIN_STATUS.FLAGS.GRP_SUSPENDED – Window's group is suspended. ?WIN_STATUS.FLAGS.WIND_SUSPENDED – Window is suspended. ?WIN_STATUS.FLAGS.KEYB_ENABLED – Keyboard input is enabled for this window. ?WIN_STATUS.FLAGS.PERM_ON – PERMANENCE is on for this window.
?WIN_STATUS.VIEW_ORG.X (double word)	X coordinate of the view port's origin (in pixels relative to the origin of the physical bit-mapped screen).
?WIN_STATUS.VIEW_ORG.Y (double word)	Y coordinate of the view port's origin (in pixels relative to the origin of the physical bit-mapped screen).
?WIN_STATUS.VIEW_SZ.X (double word)	Width of the view port (in pixels).
?WIN_STATUS.VIEW_SZ.Y (double word)	Height of the view port (in pixels).
?WIN_STATUS.SCAN_ORG.X (double word)	X coordinate of the scan port's origin (in characters for a character window, pixels for graphics window; relative to the origin of the virtual screen).
?WIN_STATUS.SCAN_ORG.Y (double word)	Y coordinate of the scan port's origin (in characters for a character window, pixels for a graphics window; relative to the origin of the virtual screen).
?WIN_STATUS.SCAN_SZ.X (double word)	Width of the scan port (in characters for a character window, pixels for a graphics window).
?WIN_STATUS.SCAN_SZ.Y (double word)	Height of the scan port (in characters for a character window, pixels for a graphics window).

(Continues)

Table 2-199. Contents of the ?WIN_WINDOW_STATUS Subpacket

Offset	Contents
?WIN_STATUS.WIND_ORG.X (double word)	X coordinate of the window's origin, including window borders, if any. (In pixels, relative to the origin of the physical bit-mapped screen.)
?WIN_STATUS.WIND_ORG.Y (double word)	Y coordinate of the window's origin, including window borders, if any. (In pixels, relative to the origin of the physical bit-mapped screen.)
?WIN_STATUS.WIND_SZ.X (double word)	Width of the window, including borders, if any. (In pixels.)
?WIN_STATUS.WIND_SZ.Y (double word)	Height of the window, including borders, if any. (In pixels.)
?WIN_STATUS.VT_SZ.X (double word)	Width of the virtual terminal (in characters for a character window, pixels for a graphics window).
?WIN_STATUS.VT_SZ.Y (double word)	Height of the virtual terminal (in characters for a character window, pixels for a graphics window).
?WIN_STATUS.VT_TYPE (double word)	Type of virtual terminal for this window. ?WIN_VT_TYPE_D460 (Character window with 2 bits per pixel.) The virtual terminal is a software emulation of a D460 terminal, and supports the blink attribute. ?WIN_VT_TYPE_MONO_D460 (Character window with 1 bit per pixel.) The virtual terminal is a software emulation of a D460 terminal, but does not support the blink attribute. ?WIN_VT_TYPE_PIXMAP (Graphics window.) The virtual terminal is a pixel map.
?WIN_STATUS.PIXEL_DEPTH	Number of bits per pixel (for graphics windows).
?WIN_STATUS.INPUT_BUF_SZ	Size of the window's input buffer, in bytes.

(Continued)

Table 2-199. Contents of the ?WIN_WINDOW_STATUS Subpacket

Offset	Contents
?WIN_STATUS.PAL_TYPE (double word)	Type of palette associated with the window. Palette types are ?WIN_PALETTE_TYPE_DEFAULT -- The system determines the palette type for this window. ?WIN_PALETTE_TYPE_TRUE -- True-color palette. Supported only on 24-bit per pixel terminals. The window cannot write to this palette. ?WIN_PALETTE_TYPE_USER -- User-definable palette with 2**(pixel depth) entries. For character windows, the pixel depth depends on the virtual terminal type. For graphics For graphics windows, the pixel depth is reported in offset ?WIN_STATUS.PIXEL_DEPTH. All entries in the palette are initially set to "dark". ?WIN_PALETTE_TYPE_SYSTEM -- System palette. The window shares the system palette, which is two entries long. The window cannot write to this palette. ?WIN_PALETTE_TYPE_D460 -- D460 palette. Has four entries, initially set to "dark," "Bright," "dim," and "blink." ?WIN_PALETTE_TYPE_MONO_D460 -- Monochrome D460 palette. Has two entries, initially set to "dark" and "bright." ?WIN_PALETTE_TYPE_SHARED -- Shared palette. The window shares the palette of the window that was specified in the main ?WINDOW packet when this window was created.

(Concluded)

Getting Window IDs

You can issue ?WINDOW with function code ?WIN_GET_WINDOW_ID to find out the window ID number of a particular window. In the main packet, specify the pathname or channel number of the window you want. There is no subpacket for the ?WIN_GET_WINDOW_ID function; set ?WIN_PKT.SUBPKT to 0. The operating system returns the window ID number in offset ?WIN_PKT.WIND_ID.

Other functions of ?WINDOW let you get the window ID numbers for all windows in a particular window group, or for all windows on a console.

Getting the Status of a Physical Device

The ?WINDOW function ?WIN_DEVICE_STATUS returns the following information about a physical device:

- The dimensions of the physical pixel map.
- The pixel depth of the physical pixel map..
- The language ID of the physical keyboard
- The type of pointer device in use (relative or absolute).

In the main packet, specify the window ID, pathname, or channel number of any window on the device you want. The operating system returns information in the subpacket. Figure 2-223 shows the structure of the subpacket; Table 2-200 details its contents.

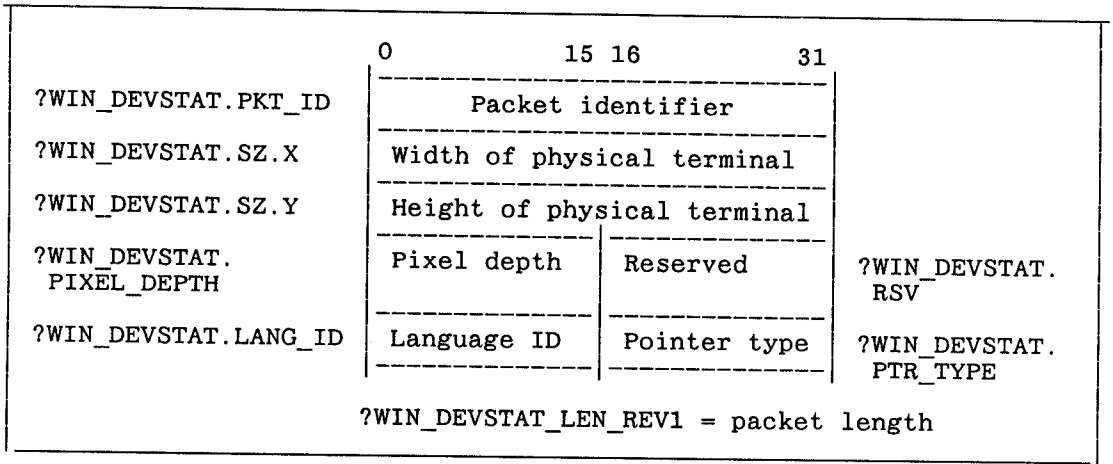


Figure 2-223. Structure of the ?WIN_DEVICE_STATUS Subpacket

Table 2-200. Contents of the ?WIN_DEVICE_STATUS Subpacket

Offset	Contents
?WIN_DEVSTAT.PKT_ID (double word)	Packet identifier; set to ?WIN_DEVSTAT_PKTID_REV1.
?WIN_DEVSTAT.SZ.X (double word)	Width of the physical screen (in pixels).
?WIN_DEVSTAT.SZ.Y (double word)	Height of the physical screen (in pixels).
?WIN_DEVSTAT. PIXEL_DEPTH	Number of bits per pixel in the physical graphics board.
?WIN_DEVSTAT.RSV	Reserved. (Set to 0.)
?WIN_DEVSTAT. LANG_ID	Language ID of the physical keyboard. Languages are ?WIN_LANG_ID_SWISS_GERMAN Swiss/German ?WIN_LANG_ID_CANADA_ENGLISH Canadian English ?WIN_LANG_ID_KATAKANA Katakana ?WIN_LANG_ID_ITALIAN Italian ?WIN_LANG_ID_CANADA_FRENCH Canadian French ?WIN_LANG_ID_US United States ?WIN_LANG_ID_UK United Kingdom ?WIN_LANG_ID_FRENCH French ?WIN_LANG_ID_GERMAN German ?WIN_LANG_ID_SWEDISH_FINNISH Swedish/Finnish ?WIN_LANG_ID_SPANISH Spanish ?WIN_LANG_ID_DANISH_NORWEGIAN Danish/Norwegian
?WIN_DEVSTAT. PTR_TYPE	Type of pointer device attached to the physical terminal. Types are ?WIN_PTR_TYPE_RELATIVE -- A relative device, such as a mouse. The pointer moves as the device moves, but the actual position of the device does not affect the location of the pointer on the screen. ?WIN_PTR_TYPE_ABSOLUTE -- An absolute device, such as a bit pad or stylus. The position of the device on a grid directly determines the location of the pointer on the screen.

To Get Window IDs for Windows in a Group

To get a list of all window IDs in a particular window group, allocate an array of memory with at least as many entries as there are windows in the group. Then, issue the ?WINDOW call with function code ?WIN_RETURN_GROUP_WINDOWS; in the main packet, specify the pathname or channel number of any window in the group you want. In the subpacket, specify the number of entries you allocated in the array. If you specify fewer entries than there are windows in the group, the system returns the number of windows in offset ?WIN_RTN_GRPWINDS.O_NUM_WINDS, and takes the error return. Figure 2-224 shows the subpacket structure.

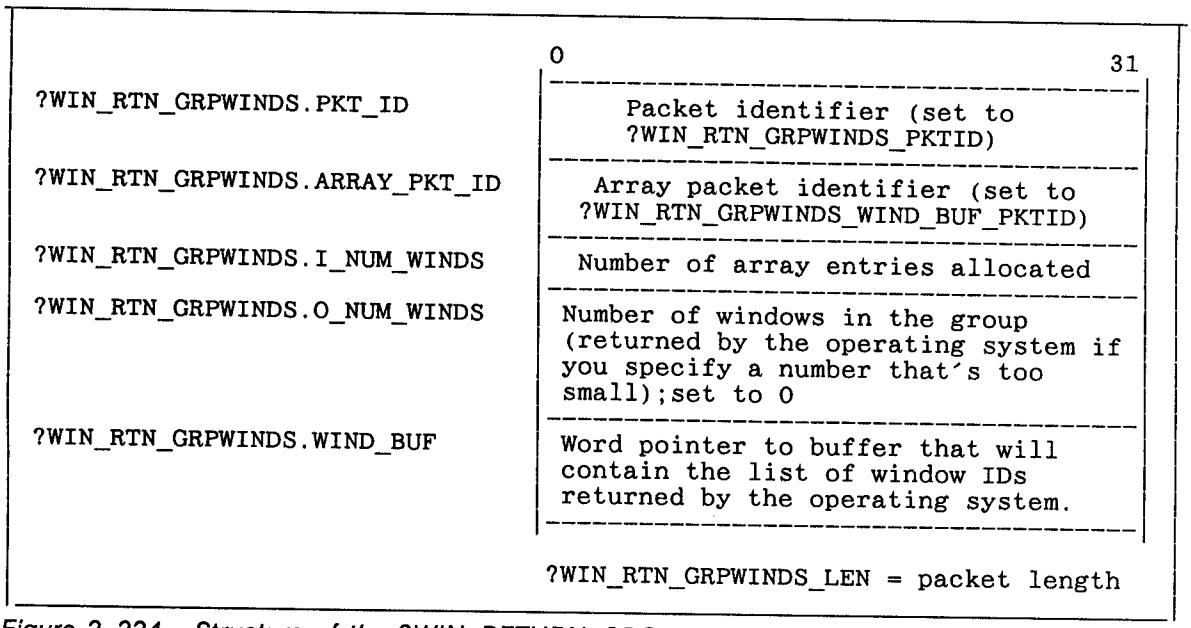


Figure 2-224. Structure of the ?WIN_RETURN_GROUP_WINDOWS Subpacket

To Get Window IDs for Windows on a Physical Terminal

To get a list of the window IDs for all the windows that exist on a particular physical terminal, allocate an array in memory, and issue the ?WINDOW call with function code ?WIN_RETURN_DEVICE_WINDOWS. In the main packet, specify the pathname or channel number of any window on the terminal you want. In the subpacket, specify how many double word entries you allocated in the array. If there are fewer entries in the array than there are windows on the terminal, the system returns the number of windows in offset ?WIN_RTN_DEVWINDS.O_NUM_WINDS, and takes the error return. Figure 2-225 shows the subpacket structure.

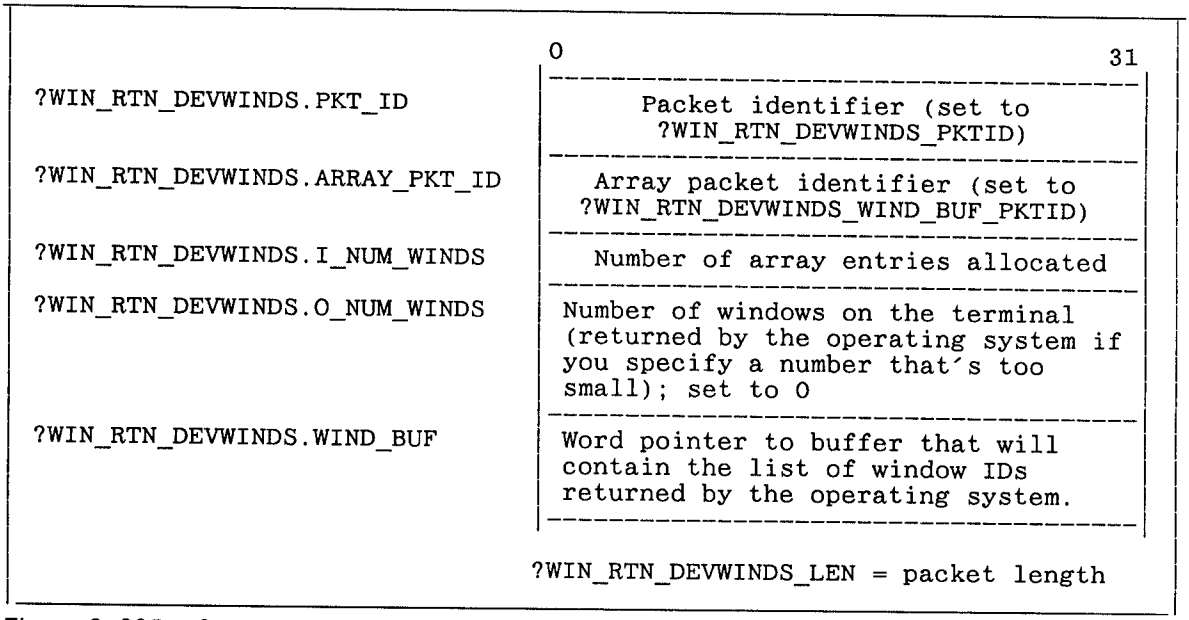


Figure 2-225. Structure of the ?WIN_RETURN_DEVICE_WINDOWS Subpacket

Writing to a Window's Palette

You use ?WINDOW function ?WIN_WRITE_PALETTE to change the colors in the palette associated with a window. You can change the color settings for one or more contiguous pixel values. For the set of pixel values you specify, you can change either the phase 0 setting, the phase 1 setting, or both. Follow these steps to change the colors in a window's palette:

1. Set up an array in memory to contain the color settings. If you are changing both the phase 0 and phase 1 settings, you must set up two separate arrays, one for each phase. The arrays must be the same size.

   There must be an entry in each array for each pixel value you want to change. An entry consists of four 32-bit unsigned fractions, one for each color level:

   <Red level> <Green level> <Blue level> <Grey level>

2. Issue the ?WINDOW call with function code ?WIN_WRITE_PALETTE. In the main packet, specify the window whose palette you are changing. In the subpacket, you specify

   - The number of palette entries you want to write.
   - The palette index at which you want to start writing.
   - The addresses of the arrays you have stored in memory.

Figure 2-226 shows the subpacket structure.

The operating system then copies the array(s) you specify onto the palette, starting at the specified palette index.

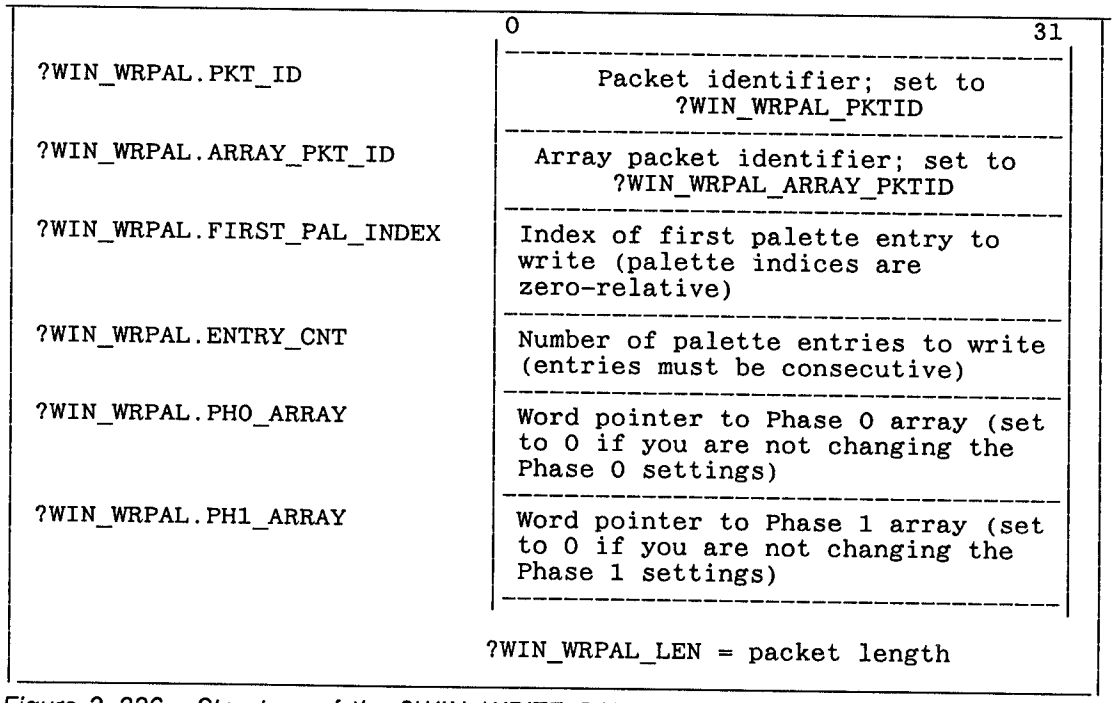


Figure 2-226. Structure of the ?WIN_WRITE_PALETTE Subpacket

Reading from a Window's Palette

You can use ?WINDOW function ?WIN_READ_PALETTE to get the current color settings from the palette associated with a window. This call copies a series of color settings for contiguous pixel values from the palette into memory. Follow these steps to determine the colors in a window's palette:

1. Allocate space in memory for each array you are copying. If you are copying both the Phase 0 and Phase 1 settings, you must allocate space for two separate arrays (one for each phase). Each array needs as much space as the number of entries you will copy, multiplied by the size of each entry (four double words).

2. Issue the ?WINDOW call with function code ?WIN_READ_PALETTE. In the main packet, specify the window whose palette you are reading. In the subpacket, you specify

   - The number of palette entries you are reading.
   - The palette index at which you want to start reading.
   - The destination memory address for each array.

The operating system then copies the palette entries you specified into the space you allocated in memory. The copied entries are set up as arrays, one for each phase. Each array contains an entry for each pixel value you requested. Each entry consists of four double words, each of which denotes a color level:

<Red level> <Green level> <Blue level> <Grey level>

Figure 2-227 shows the subpacket structure.

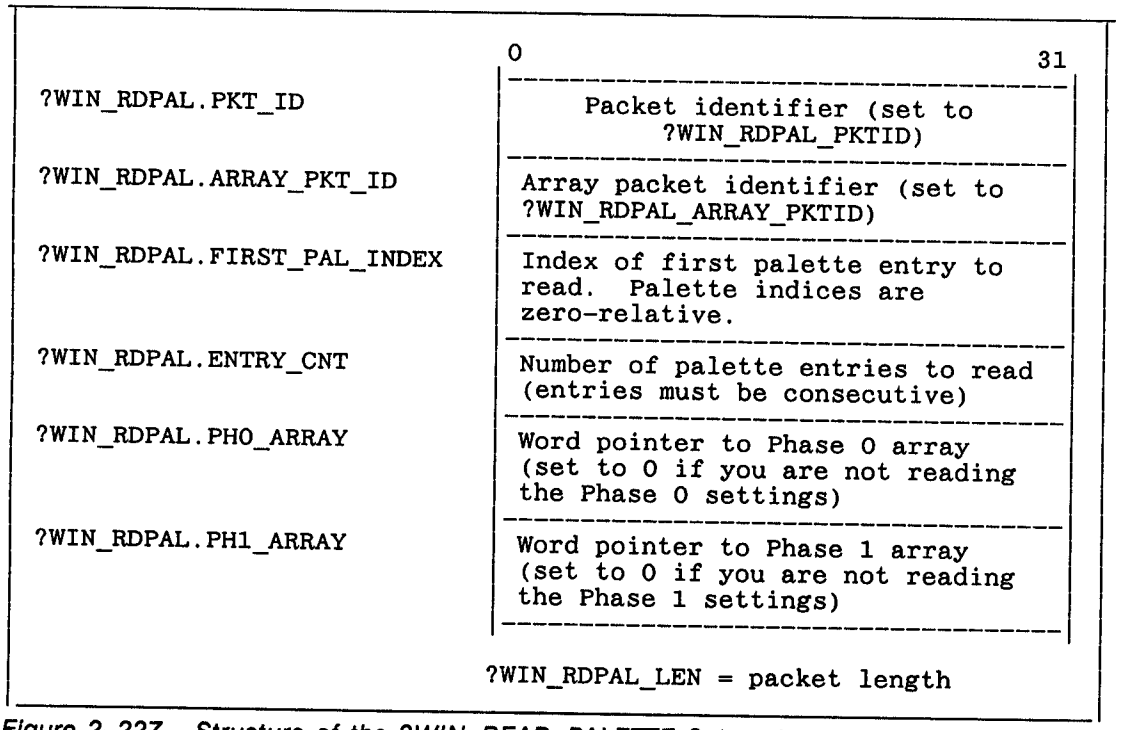


Figure 2-227. Structure of the ?WIN_READ_PALETTE Subpacket

?WIRE

Wires pages to the working set.

?WIRE
error return
normal return

Input

AC0	Starting address of the region to wire
AC1	Ending address of the region to wire
AC2	Reserved (Set to 0.)

Output

AC0	Unchanged
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERAWM	Attempt to wire more pages than working set maximum.
ERICM	Illegal system command (invalid call for 16-bit processes)
ERPRE	Invalid system call parameter
ERPRV	Caller not privileged for this action.
ERPTY	Illegal process type
ERTPW	Too many pages wired. (The number of wired pages exceeds the limits of the swap file allocated to this process. Allowing another wired page would prevent swapping.)
ERVWP	Invalid word pointer passed as a system call argument

Why Use It?

?WIRE prevents the operating system from removing a range of logical pages from your working set. The pages that you wire with ?WIRE remain in your working set until you issue an ?UNWIRE call for them.

What It Does

?WIRE permanently binds (that is, wires) the region that you specify in AC0 and AC1 to the caller's working set. The operating system adjusts the caller's current working set size accordingly.

You can issue more than one ?WIRE against the same region, but subsequent system calls have no additional effect.

Before you issue ?WIRE, load AC0 and AC1 with the starting and ending addresses, respectively, of the region you want to wire. ?WIRE affects all pages within the region that you specify, including the pages that contain the starting and ending addresses. Note that this differs from the interpretation of the target region for ?UNWIRE.

There need not be a one-to-one correspondence between the ?WIRE and ?UNWIRE system calls that you issue. In other words, you can issue several ?WIRE system calls for different address regions, and then issue one ?UNWIRE system call to revoke all the previous ?WIRE system calls.

Notes

- See the description of ?UNWIRE in this chapter.

?WRB

Writes block I/O.

?WRB *[packet address]*
error return
normal return

See the description of ?RDB in this chapter for more information on ?WRB.

Licensed Material - Property of Data General Corporation

?WRITE

Writes record I/O.

?WRITE *[packet address]*
error return
normal return

See the description of ?READ in this chapter for more information on ?WRITE.

?WRUDA

Writes a user data area (UDA).

?WRUDA
error return
normal return

See the description of ?RDUDA in this chapter for more information on ?WRUDA.

Licensed Material - Property of Data General Corporation

?WTSIG

Waits for a signal from another task or process.

?WTSIG
error return
normal return

Input

None

Output

AC0	Undefined
AC1	Unchanged
AC2	Undefined

Error Codes in AC0

ERSCI	System call at interrupt level

Why Use It?

?WTSIG, along with ?SIGWT and ?SIGNL, allows tasks to synchronize access to global memory that identical local servers share.

?WTSIG Continued

What It Does

?WTSIG is similar to the IPC system call ?IREC. However, unlike ?IREC, ?WTSIG does not cause data movement. Instead, ?WTSIG waits for a signal (?SIGNL or ?SIGWT) from one of the following:

- Another task in the same process.
- Another task in a different process.

Warnings

The fast inter-process signalling mechanism is used by Data General supplied software, including (but not limited to) the PMGR interface routines found in the Agent in Ring 3 of all processes in the operating system. And, the system makes absolutely no checks as to the legitimacy of a ?SIGNL issued by anyone or to anyone.

These two warnings have the following implications.

If a task issues a ?SIGNL to another task, it must be absolutely sure that the next ?SIGWT or ?WTSIG issued by the receiving task will be the one it expects to receive the ?SIGNL. If such is not the case, the ?SIGNL may go to a ?SIGWT or ?WTSIG issued by another routine used by the task elsewhere in the process (e.g., the PMGR interface in Ring 3, or the INFOS® local server in Ring 4). There is no addressing of the message as is provided by the IPC message port number or the ?REC/?XMT mailbox location.

Then, after any task resumes execution after receipt of a ?SIGNL by ?WTSIG or ?SIGWT, it is incumbent upon the receiver to verify that the condition being waited for has in fact occurred. The operating system does not authenticate ?SIGNLs sent.

Notes

- See the descriptions of ?IREC, ?SIGNL, and ?SIGWT in this chapter.

Licensed Material - Property of Data General Corporation

?XINIT — Initializes a logical disk (extended).

AOS/VS only

?XINIT *[packet address]*error return normal return

Input

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

Output

AC0	Unchanged
AC1	Unchanged
AC2	Address of the ?XINIT packet

Error Codes in AC0

ERCNM	Controller does not support logical disk unit (LDU) mirroring
EREAD	Execute access denied
ERFIX	Can't initialize LD, run FIXUP on it
ERIDD	Inconsistent disk indentification block DIB data (Disk is not a valid operating system disk type.)
ERIDU	Incomplete logical disk (LD)
ERILD	Inconsistent LD
ERIML	Incomplete mirrored LDU specified
ERLFM	LDU format mismatch – not a valid LDU mirror
ERLMM	LDU name mismatch – not a valid LDU mirror
ERLSZ	LDU size mismatch – not a valid LDU mirror
ERMIS	Disk and file system revision numbers don't match
ERMSY	Mirrored LDU is not synchronized
ERNAM	LDU is not mirrored
ERRID	LDU IDs are not unique – not a valid LDU mirror
ERSIP	Cannot initialize an LDU image that was previously being synchronized
ERVBP	Invalid byte pointer passed as a system call argument
ERVWP	Invalid word pointer passed as a system call argument

?XINIT Continued

Why Use It?

You use ?XINIT (or ?INIT) to initialize a logical disk. In addition, you can use ?XINIT to initialize a set of synchronized LDU images.

What It Does

Initializing a set of synchronized LDU images via ?XINIT means that you intend to include at least one mirrored LDU into the existing file structure. If one LDU image becomes unavailable, you still have access to the LDU data on the remaining available image.

?XINIT sizes the disk units, creates memory databases, and recalibrates the disk.

The caller's privileges must include Owner access to the target LD's root. These privileges must also include either Write or Append access to the target directory. In addition, the caller must have Execute access to all the physical disks specified. ?XINIT's privileges are the same ones that ?INIT must have.

?XINIT is an extended version of, and upward compatible with, ?INIT. We recommend that you use ?XINIT instead of ?INIT in all new programs that you write. There is no need for an extended release call since ?RELEASE has all the functionality required to handle a logical disk that either ?INIT or ?XINIT has initialized. Furthermore, you issue ?MIRROR to synchronize mirrored images that ?XINIT cannot initialize and to manage mirrored LDU configurations.

The main packet contains pointers to subpackets that contain physical unit lists for each LDU image. You specify two images when you initialize a synchronized set of images (for mirrored LDUs). Figure 2–228 shows the structure of the ?XINIT main packet, and Table 2–201 describes its contents. Figure 2–229 shows the structure and contents of the ?XINIT subpacket. This subpacket is identical to the one that ?MIRROR uses except for the value of the packet identifier.

Licensed Material - Property of Data General Corporation

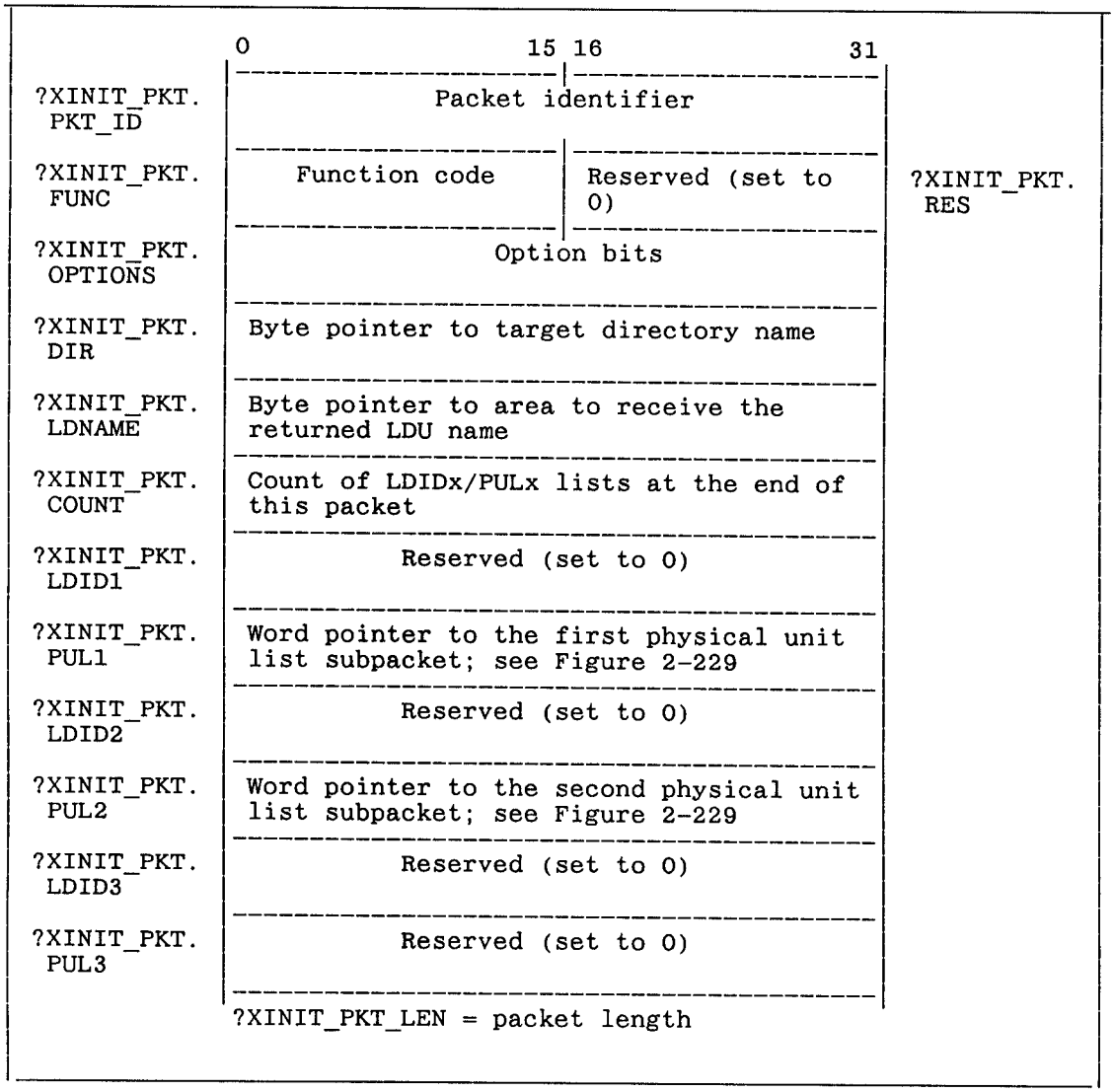


Figure 2-228. Structure of ?XINIT Packet

?XINIT Continued

Table 2-201. Contents of ?XINIT Packet

Offset	Contents
?XINIT_PKT.PKT_ID (double word)	Packet identifier. Place ?XINIT_PKT_PKTID here.
?XINIT_PKT.FUNC	Function code. Place one of the following values here: - ?XINIT_WORKING_DIR to graft a LDU into the working directory. - ?XINIT_ROOT to graft a LDU into the root directory (:). - ?XINIT_TARGET_DIR to graft a LDU into the target directory that you specify in offset ?XINIT_PKT.DIR.
?XINIT_PKT.RES	Reserved. (Set to 0.)
?XINIT_PKT.OPTIONS (double word)	Options word. Place one of the following values here: o ?XINIT_OVERRIDE to override the error condition ERIML ("Incomplete mirrored LDU specified"). The operating system might return the following values here: o ?XINIT_FIXUP_REC to recommend that you run utility program FIXUP.
?XINIT_PKT.DIR (double word)	If you specified option ?XINIT_TARGET_DIR in offset ?XINIT_PKT.FUNC, then offset ?XINIT_PKT.DIR must contain a byte pointer to the name of the directory into which the LDU will be grafted. In turn, this name must end with a null byte. If you didn't specify option ?XINIT_TARGET_DIR in offset ?XINIT_PKT.FUNC, then supply 0 here.
?XINIT_PKT.LDNAME (double word)	Supply a byte pointer to the area where the system call returns the LDU name. This area must be ?MXFN bytes long.
?XINIT_PKT.COUNT (double word)	Supply the nonzero number of LDIDx/PULx (where x is 1 or 2) pairs of offsets that you supply next in this parameter packet. The number cannot exceed ?XINIT_MAXIMAGES.
?XINIT_PKT.LDID1	Reserved. (Set to 0.)

(continues)

Table 2-201. Contents of ?XINIT Packet

Offset	Contents
?XINIT_PKT.PUL1 (double word)	Supply a word pointer to the first physical unit list subpacket. Figure 2-229 describes this subpacket.
?XINIT_PKT.LDID2	Reserved. (Set to 0.)
?XINIT_PKT.PUL2 (double word)	Supply, if necessary (i.e., for mirroring), a word pointer to the second physical unit list subpacket. Figure 2-229 describes this subpacket. If you don't need the subpacket, supply 0.
?XINIT_PKT.LDID3	Reserved. (Set to 0.)
?XINIT_PKT.PUL3 (double word)	Reserved. (Set to 0.)

(Concluded)

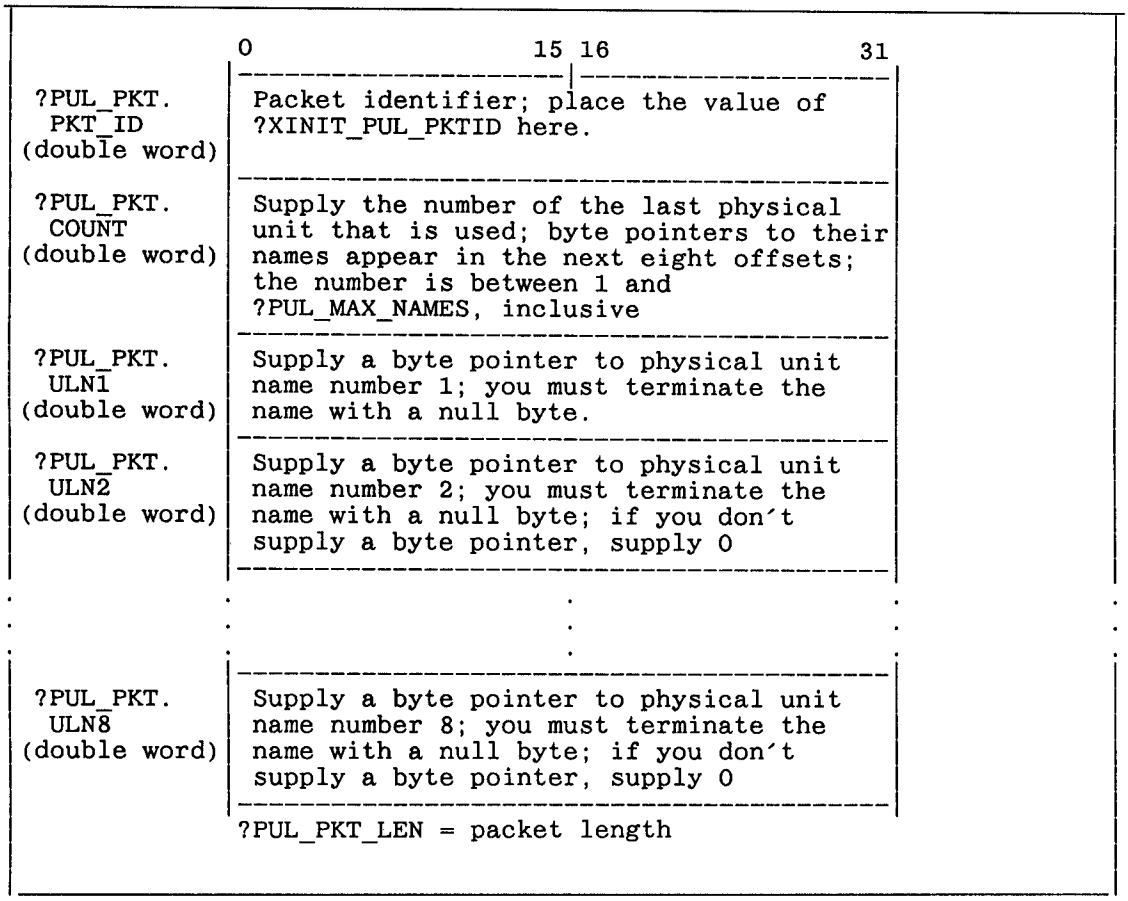


Figure 2-229. Structure of ?XINIT Subpacket

16-bit Version of ?XINIT System Call

A 16-bit version of this system call also exists. Its functionality is identical to that of the 32-bit version previously described.

Figure 2-230 contains the structure of the 16-bit ?XINIT call main packet and Figure 2-231 contains the structure of the 16-bit ?XINIT call subpacket. The definitions of the offsets in these packets are identical to those in Figure 2-228 and Figure 2-229, respectively. The only differences in the corresponding packets are the names of the offsets.

Table 2-202 describes the contents of Figure 2-230.

Notes

- See the descriptions of ?RELEASE, ?MIRROR, and ?XINIT in this chapter.

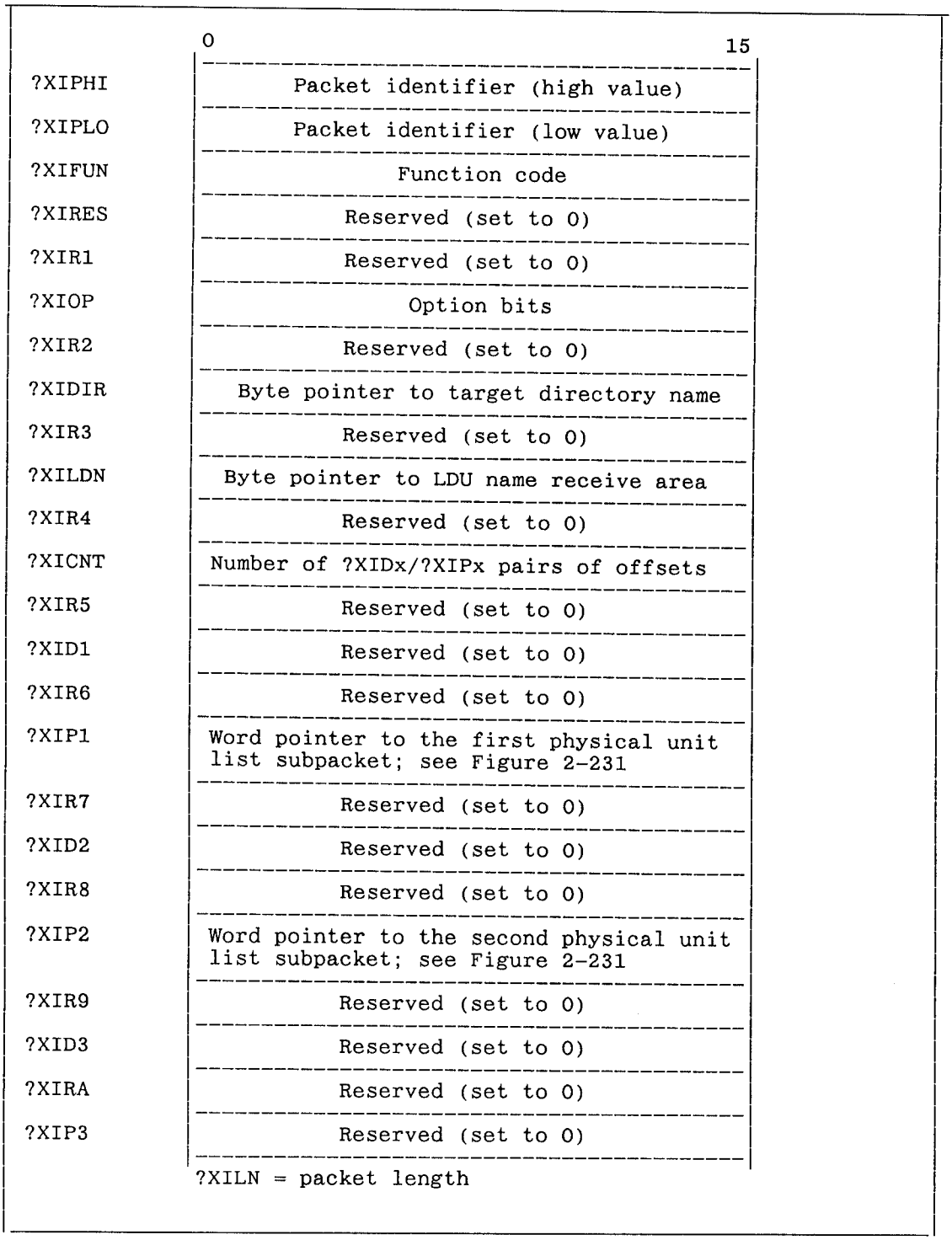


Figure 2-230. Structure of 16-Bit ?XINIT Packet

Table 2-202. Contents of 16-Bit ?XINIT Packet

Offset	Contents
?XIPHI	Packet identifier (high word). Place ?XII1 here.
?XIPLO	Packet identifier (low word). Place ?XII2 here.
?XIFUN	Function code. Place one of the following values here: - ?XFGWD to graft a LDU into the working directory. - ?XFGRT to graft a LDU into the root directory (:). - ?XIGTD to graft a LDU into the target directory that you specify in offset ?XIDIR.
?XIRES	Reserved. (Set to 0.)
?XIR1	Reserved. (Set to 0.)
?XIOP	Options word. Place one of the following values here: - ?XBORD to override the error condition ERIML ("Incomplete mirrored LDU specified"). The operating system might return the following values here: - ?XBFXR to recommend that you run utility program FIXUP.
?XIR2	Reserved. (Set to 0.)
?XIDIR	If you specified ?XIGTD in offset ?XIFUN, then offset ?XIDIR must contain a byte pointer to the name of the directory into which the LDU will be grafted. In turn, this name must end with a null byte. If you didn't specify option ?XIGTD in offset ?XIFUN, then supply 0 here.
?XIR3	Reserved. (Set to 0.)
?XILDN	Supply a byte pointer to the area where the system call returns the LDU name. This area must be ?MXFN bytes long.
?XIR4	Reserved. (Set to 0.)

(Continues)

Table 2-202. Contents of 16-Bit ?XINIT Packet

Offset	Contents
?XICNT	Supply the nonzero number of ?XIDx/?XIPx (where x is 1 or 2) pairs of offsets that you supply next in this parameter packet. The number cannot exceed ?XINIT_MAXIMAGES.
?XIR5	Reserved. (Set to 0.)
?XID1	Reserved. (Set to 0.)
?XIR6	Reserved. (Set to 0.)
?XIP1	Supply a word pointer to the first physical unit list subpacket. Figure 2-231 describes this subpacket.
?XIR7	Reserved. (Set to 0.)
?XID2	Reserved. (Set to 0.)
?XIR8	Reserved. (Set to 0.)
?XIP2	Supply, if necessary (i.e., for mirroring), a word pointer to the second physical unit list subpacket. Figure 2-231 describes this subpacket. If you don't need the subpacket, supply 0.
?XIR9	Reserved. (Set to 0.)
?XID3	Reserved. (Set to 0.)
?XIRA	Reserved. (Set to 0.)
?XIP3	Reserved. (Set to 0.)

(Concluded)

	0 15
?ULPHI	Packet identifier (high word); place the value of ?ULI1 here
?ULPLO	Packet identifier (low word); place the value of ?ULI3 here
?ULRO	Reserved (set to 0)
?ULNC	Supply the number of the last physical unit that is used; byte pointers to their names appear in the next sixteen offsets; the number is between 1 and ?MAXNAMES, inclusive
?ULR1	Reserved (set to 0)
?ULN1	Supply a byte pointer to physical unit name number 1; you must terminate the name with a null byte.
?ULR2	Reserved (set to 0)
?ULN2	Supply a byte pointer to physical unit name number 2; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0
?ULR3	Reserved (set to 0)
?ULN3	Supply a byte pointer to physical unit name number 3; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0
?ULR4	Reserved (set to 0)
?ULN4	Supply a byte pointer to physical unit name number 4; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0
?ULR5	Reserved (set to 0)
?ULN5	Supply a byte pointer to physical unit name number 5; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0
?ULR6	Reserved (set to 0)
?ULN6	Supply a byte pointer to physical unit name number 6; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0
?ULR7	Reserved (set to 0)
?ULN7	Supply a byte pointer to physical unit name number 7; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0
?ULR8	Reserved (set to 0)
?ULN8	Supply a byte pointer to physical unit name number 8; you must terminate the name with a null byte; if you don't supply a byte pointer, supply 0

?ULLN = packet length

Figure 2-231. Structure of 16-Bit ?XINIT Subpacket

Licensed Material - Property of Data General Corporation

?XMT — Transmits an intertask message.

?XMT
error return
normal return

Input

AC0	Address of the mailbox that is to receive the intertask message
AC1	Intertask message
AC2	To broadcast the intertask message to all waiting, -1

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERVWP	Invalid word pointer passed as a system call argument
ERXMT	Signal to address already in use (You tried to send a message to a mailbox that already contains a message.)
ERXMZ	Attempt to XMT illegal message

Why Use It?

?XMT is one of four system calls for intertask communications. (The others are ?XMTW, ?REC, and ?RECNW.) You can use the intertask communications facility to pass variables and data or to synchronize tasks.

What It Does

?XMT sends a nonzero message to the mailbox that you specify in AC0. The mailbox must contain 0 before you issue ?XMT.

Before you issue ?XMT, load AC0 with the address of the mailbox, and load AC1 with the message. The receiving task gets the message by issuing either a ?REC or a ?RECNW against the same mailbox. (For 32-bit processes, the mailboxes are 32 bits wide; for 16-bit processes, the mailboxes are 16 bits wide.)

The "broadcast" option allows you to send the message to all other tasks that issue corresponding ?REC or ?RECNW system calls. If you do not broadcast the message and more than one task has issued a receive, the message goes to the highest priority receiver. If all waiting receivers have the same priority, the task scheduler selects the receiver.

Note that ?XMT may cause immediate task rescheduling if there is a waiting receiver with a higher priority than the ?XMT caller.

Notes

- See the descriptions of ?XMTW, ?REC, and ?RECNW in this chapter.

?XMTW — Transmits an intertask message and waits for it to be received.

?XMTW
error return
normal return

Input

AC0	Address of the mailbox that is to receive the intertask message
AC1	Message
AC2	To broadcast the message to all waiting receivers, -1

Output

AC0	Unchanged
AC1	Unchanged
AC2	Unchanged

Error Codes in AC0

ERVWP	Invalid word pointer passed as a system call argument
ERXMT	Signal to address already in use (You tried to send a message to a mailbox that already contains a message.)
ERXMZ	Attempt to XMT illegal message

Why Use It?

You can use ?XMTW to synchronize the calling task with one or more receiving tasks. ?XMTW is more appropriate than ?XMT for this purpose, because it suspends the calling task until the receiver picks up the message.

What It Does

Like ?XMT, ?XMTW allows the calling task to send a nonzero message to an empty mailbox and, ultimately, to the task or tasks that issue ?REC or ?RECNW system calls against that mailbox. (For 32-bit processes, the mailboxes are 32 bits wide; for 16-bit processes, the mailboxes are 16 bits wide.)

?XMTW differs from ?XMT in that it suspends the calling task until a receiving task issues either ?REC or ?RECNW against the mailbox. The operating system does not suspend the ?XMTW caller if there is an outstanding receive against the mailbox.

?XMTW takes the same input parameters as ?XMT. Note that ?XMTW also allows you to "broadcast" the message to all waiting receivers, by loading AC2 with -1 at input.

?XMTW may cause immediate task rescheduling if a waiting receiver has higher priority than the ?XMTW caller.

Notes

- See the descriptions of ?XMTW, ?REC, and ?RECNW in this chapter.

?XPSTAT

Returns extended status information on a process.

?XPSTAT *[packet address]*
error return
normal return

Operating System Differences	
Accumulator	
Input and Output	Some
Error Codes	None
Parameter Packet	Some

Input

AC0 One of the following:

- PID or VPID of the target process
- Byte pointer to the name of the target process
- −1 for status information about the calling process

AC1 One of the following:

- −1 if AC0 is a byte pointer
- Any other value if AC0 contains either −1, a PID, or a VPID

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

Output

AC0 Unchanged

AC1 Unchanged

AC2 Address of the ?XPSTAT packet

Under AOS/DVS, the target process name that AC0 points to may contain a process group name. An example is P_GROUP:USER_NAME:PROC_NAME. Consequently, you may identify remote processes either by name or by ID number (VPID).

Error Codes in AC0

ERIRB	Insufficient room in buffer (for username, ?PROC-related username, memory descriptors, or process group name)
ERPNM	Illegal process name
ERPRH	Attempt to access process not in hierarchy
ERPVS	Packet revision not supported
ERVBP	Illegal byte pointer in AC0
ERVWB	Packet failed read/write validation
ERVWP	Illegal word pointer in AC2

Why Use It?

Like ?PSTAT, ?RUNTM, and ?WHIST, ?XPSTAT helps you to determine how well a process competes with others of the same type for system resources, such as memory and CPU time. Note that the operating system updates most of the ?XPSTAT information as the process executes.

What It Does

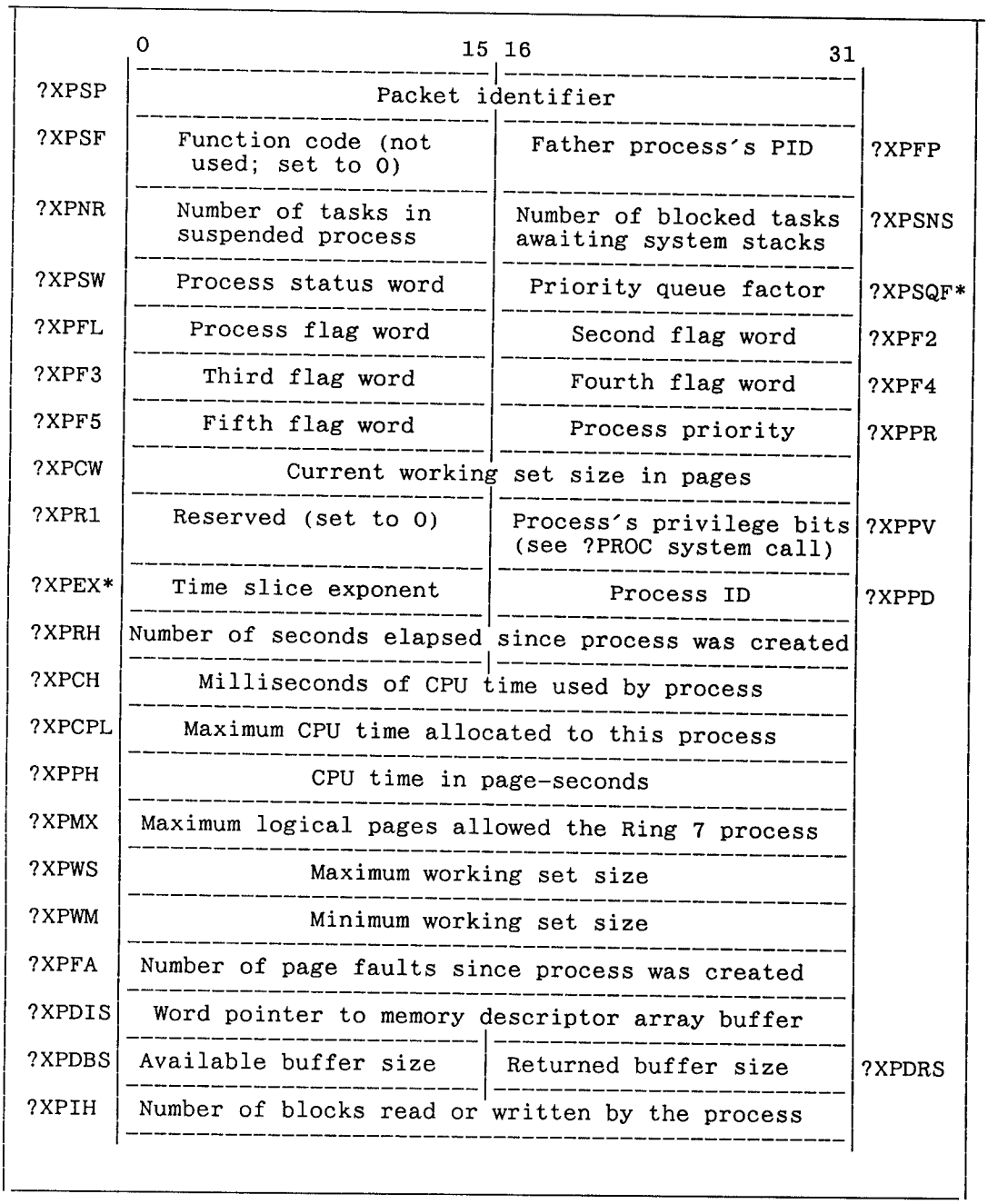
?XPSTAT returns internal statistics about the calling process or about any process the caller specifies in AC0. The operating system returns the statistics in the packet. Figure 2-232 shows the structure and Table 2-203 describes the contents of the ?XPSTAT packet.

In addition to the process statistics, the operating system returns seven sets of memory descriptors in the packet, one descriptor for each of the rings 1 through 7. Each descriptor contains information about the program currently running in that ring. Figure 2-233 shows the memory descriptor structure. It is identical to Figure 2-174.

?XPSTAT provides most of the information that ?PSTAT provides. Only the offsets from ?PSSN through ?PSEN (?PSAL words) are in ?PSTAT alone. These words return information about son PIDs from 1 through 255. However, system call ?SONS provides you with the information in these offsets — and with information about son PIDs greater than 255.

?XPSTAT also provides you with information in the ?PROC system call's extended packet, such as current user locality and legal localities. So, we recommend that you use ?XPSTAT and/or ?SONS in your new programs since (together) they have all the functionality of ?PSTAT — and more. ?XPSTAT also executes considerably faster than ?PSTAT.

Parameter Packet



* This offset differs between AOS/VS and AOS/DVS. See Table 2-203.

Figure 2-232. Structure of ?XPSTAT Packet (Continues)

?XPSTAT Continued

Parameter Packet

	0 — 15	16 — 31	
?XPLFA	Number of page faults that did not require disk I/O		
?XPSL*	Number of subslices left	CPU number	?XPCPU*
?XPLL*	Legal localities	User locality	?XPULC*
?XPPL*	Program locality	Class ID of process	?XPCID*
?XPPG*	Byte pointer to process group name		
?XPGBS*	Available buffer size	Returned buffer size	?XPGRS*
?XPUN	Byte pointer to username		
?XPNBS	Available buffer size	Returned buffer size	?XPNRS
?XPPU	Byte pointer to ?PROC-related username		
?XPUBS	Available buffer size	Returned buffer size	?XPURS
?XPUPD*	1st word of 128-bit unique PID	2nd word of 128-bit unique PID	
	3rd word of 128-bit unique PID	4th word of 128-bit unique PID	
	5th word of 128-bit unique PID	6th word of 128-bit unique PID	
	7th word of 128-bit unique PID	8th word of 128-bit unique PID	

?XPLTH = packet length

* This offset differs between AOS/VS and AOS/DVS. See Table 2-203.

Figure 2-232. Structure of ?XPSTAT Packet (Concluded)

Licensed Material - Property of Data General Corporation

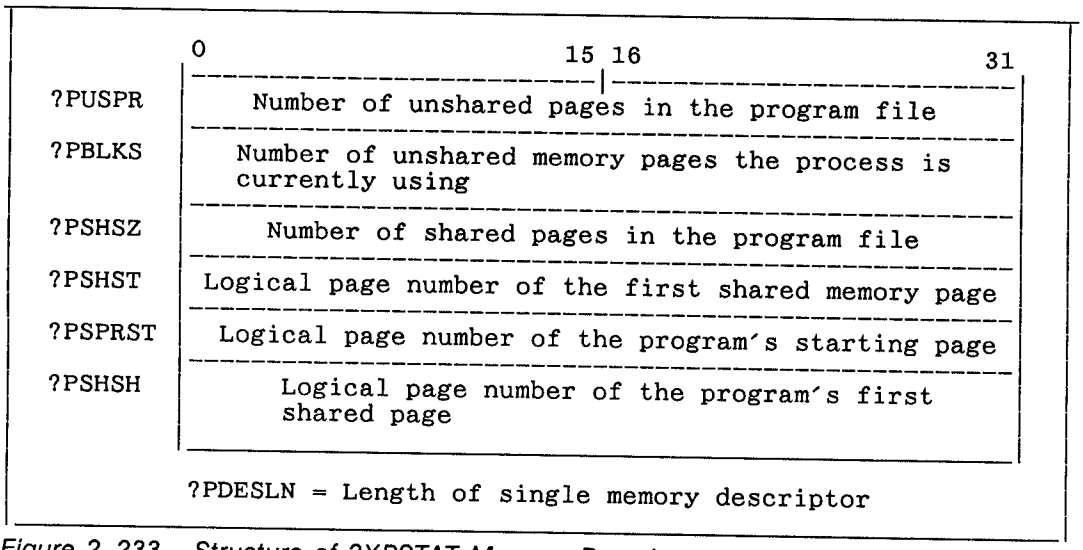


Figure 2-233. Structure of ?XPSTAT Memory Descriptor

?XPSTAT Continued

In Table 2–203, the column heads "VS" and "DVS" represent "AOS/VS" and "AOS/DVS" respectively. The entries in these columns have the following codes:

---	An operating system does not define a value for the given offset.
U	An operating system handles the offset differently (uniquely).
[no entry]	The operating systems handle the offset in the same way.

Table 2–203. Contents of ?XPSTAT Parameter Packet

Offset	VS	DVS	Description
?XPSP (double word)			Packet identifier. Place ?XPSID here.
?XPSF			Function code. Place 0 here.
?XPFP			PID or VPID of the process's parent process. The operating systems reserve a 16-bit word for the small PID. User programs that mask the value returned to 8 bits (length of "low pids") will not return correct PID values for creating processes with hybrid- or anyPIDs.
?XPNR			Number of tasks in the process currently suspended on ?IREC system calls.
?XPSNS			Number of tasks that are blocked awaiting system stacks.
?XPSW			Process status word (Bit 7: process is blocked).
?XPSQF		---	Priority queue factor (indicates the process's priority for execution relative to other processes of the same type). AOS/DVS does not return this information.
?XPFL			Process flag word. ?PSSP--process is swappable. ?PSPP--process is pre-emptible. ?PSTA--type A process. ?PSTB--type B process. ?PSTC--type C process.
?XPF2			Second flag word.

(Continues)

Table 2-203. Contents of ?XPSTAT Parameter Packet

Offset	VS	DVS	Description
?XPF3			Third flag word.
?XPF4			Fourth flag word.
?XPF5			Fifth flag word.
?XPPR			Process priority.
?XPCW (double word)			Current number of shared and unshared pages in process's logical address space (the OS user-defined "working set" size).
?XPR1			Reserved. (Set to 0.)
?XPPV			Process's privilege bits. (See ?PROC in this system call dictionary for information on access privileges.)
?XPEX	U	U	Current time slice exponent (value indicating the amount of CPU time currently allocated to the target process). Systems determine value heuristically using "subslice," a constant, and "s", the time-slice exponent: AOS/VS: (time slice exponent range of 1 - 6) time slice = subslice * 2**s AOS/DVS: (time slice exponent range of X - X) time slice = subslice * XXX (See offset ?XPSL in this table.)
?XPPD			PID or VPID of the process.
?XPRH (double word)			Number of seconds that have elapsed since the process was created.
?XPCH (double word)			CPU time (in milliseconds) used by the process.
?XPCPL (double word)			Maximum CPU time allocated to this process.
?XPPH (double word)			CPU time (in page-seconds).
?XPMX (double word)			Maximum number of logical pages allowed the process in Ring 7.

(Continued)

Table 2-203. Contents of ?XPSTAT Parameter Packet

Offset	VS	DVS	Description
?XPWS (double word)			Maximum working set size in pages.
?XPWM (double word)			Minimum working set size in pages.
?XPFA (double word)			Number of page faults since the process was created (with ?PROC).
?XPDIS (double word)			Word pointer to a buffer that receives the memory descriptor array. See Figure 2-233. If you supply -1, the operating system returns nothing in the buffer.
?XPDBS			Size of the buffer, in words, that ?XPDIS points to. This number must be at least 7*?PDESLN.
?XPDRS			Actual number of words returned in the buffer that ?XPDIS points to. This is 7*?PDESLN.
?XPIH (double word)			Number of blocks read or written by the process.
?XPLFA (double word)			Number of page faults that did not require disk input/output.
?XPSL	U	U	Number of remaining subslices in the process's time slice. This value is based on the current time slice exponent returned in ?XPEX. If the time slice exponent is 1, an AOS/VS process has 2 remaining subslices (that is, 2**1). AOS/DVS does not break time slices into subslices.
?XPCPU		---	CPU number. AOS/DVS does not return this number.
?XPLL		---	Legal localities. AOS/DVS does not return this number.
?XPULC		---	Current user locality. AOS/DVS does not return this number.
?XPPL		---	Program locality. AOS/DVS does not return this number.
?XPCID		---	Class ID of the process. AOS/DVS does not return this number.
?XPPG (double word)	---		Byte pointer to the buffer that receives the process group name. If you supply -1, AOS/DVS returns nothing in the buffer. AOS/VS does not return this information.

(Continued)

Table 2-203. Contents of ?XPSTAT Parameter Packet

Offset	VS	DVS	Description
?XPGBS	---		Number of bytes in the buffer that ?XPPG points to. It must be at least ?MXHN, unless you supply -1 in offset ?XPPG. AOS/VS does not return this number.
?XPGRS	---		Actual number of bytes returned in the buffer that ?XPPG points to. AOS/VS does not return this information.
?XPUN (double word)			Byte pointer to the buffer that receives the username. If you supply -1, the operating system returns nothing in the buffer.
?XPNBS			Number of bytes in the buffer that ?XPUN points to. It must be at least ?MXUN, unless you supply -1 in offset ?XPUN.
?XPNRS			Actual number of bytes returned in the buffer (including the terminating null) that ?XPUN points to.
?XPPU (double word)			Byte pointer to the buffer that receives the ?PROC-related username. If you supply -1, the operating system returns nothing in the buffer.
?XPUBS			Number of bytes in the buffer that ?XPPU points to. It must be at least ?MXUN, unless you supply -1 in offset ?XPPU.
?XPURS			Actual number of bytes returned in the buffer (including the terminating null) that ?XPPU points to.
?XPUPD (8 words)	---		The operating system returns the process's 128-bit unique PID. AOS/VS returns nothing.

(Concluded)

Notes

- See the descriptions of ?PSTAT, ?RUNTM, and ?WHIST in this chapter.
- The parameters shown in Figure 2–233 are offsets from the memory descriptor base address. To get the base address for a particular ring, use the following formula:

  <address_returned_in_offset_?XPDIS> + ?PMDIS + (?PDESLN * (ring_number–1))

  where

  ?XPDIS — is the offset in the ?PSTAT packet that contains the address of the memory descriptors.

  ?PDESLN — is the length of each memory descriptor.

  Assume, for example, that you loaded a program file into Ring 5 with the ?RINGLD system call. To obtain the base address of the descriptor for that ring, you would use the following formula:

  <address_returned_in_offset_?XPDIS> + ?PMDIS + (?PDESLN * 4)

End of Chapter

A

Error Codes

When a system call takes an error return, it places an error code in AC0. Each error code has an octal value and a mnemonic that represents the code. In addition, each error code has a text message associated with it. This appendix lists the error codes and their text messages in two ways: in numeric order of octal values (Table A–1) and in alphabetical order of mnemonics (Table A–2).

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
1	ERICM	ILLEGAL SYSTEM COMMAND
2	ERFNO	CHANNEL NOT OPEN
3	EROPR	CHANNEL ALREADY OPEN
4	ERSAL	SHARED I/O REQ NOT MAP SLOT ALIGNED
5	ERMEM	INSUFFICIENT MEMORY AVAILABLE
6	ERADR	ILLEGAL STARTING ADDRESS
7	EROVN	ILLEGAL OVERLAY NUMBER
10	ERTIM	ILLEGAL TIME ARGUMENT
11	ERNOT	NO TASK CONTROL BLOCK AVAILABLE
12	ERXMT	SIGNAL TO ADDRESS ALREADY IN USE
13	ERQTS	ERROR IN QTASK REQUEST
14	ERTID	TASK I.D. ERROR
15	ERDCH	DATA CHANNEL MAP FULL
16	ERMPR	SYSTEM CALL PARAMETER ADDRESS ERROR
17	ERABT	TASK NOT FOUND FOR ABORT
20	ERIRB	INSUFFICIENT ROOM IN BUFFER
21	ERSPC	FILE SPACE EXHAUSTED
22	ERSFT	USER STACK FAULT
23	ERDDE	DIRECTORY DOES NOT EXIST
24	ERIFC	ILLEGAL FILENAME CHARACTER
25	ERFDE	FILE DOES NOT EXIST
26	ERNAE	FILE NAME ALREADY EXISTS
27	ERNAD	NON-DIRECTORY ARGUMENT IN PATHNAME
30	EREOF	END OF FILE
31	ERDID	DIRECTORY DELETE ERROR
32	ERWAD	WRITE ACCESS DENIED
33	ERRAD	READ ACCESS DENIED
34	ERAWD	APPEND AND/OR WRITE ACCESS DENIED
35	ERNMC	NO CHANNELS AVAILABLE
36	ERSRL	RELEASE OF NON-ACTIVE SHARED SLOT
37	ERPRP	ILLEGAL PRIORITY
40	ERBMX	ILLEGAL MAX SIZE ON PROCESS CREATE
41	ERPTY	ILLEGAL PROCESS TYPE
42	ERCON	CONSOLE DEVICE SPECIFICATION ERROR
43	ERNSW	SWAP FILE SPACE EXHAUSTED
44	ERIBS	DEVICE ALREADY IN SYSTEM
45	ERDNM	ILLEGAL DEVICE CODE
46	ERSHP	ERROR ON SHARED PARTITION SET
47	ERRMP	ERROR ON REMAP CALL
50	ERGSG	ILLEGAL AGENT GATE CALL

(Continues)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
51	ERPRN	NUMBER OF PROCESSES EXCEEDS MAX
52	ERNEF	IPC MESSAGE EXCEEDS BUFFER LENGTH
53	ERIVP	INVALID PORT NUMBER
54	ERNMS	NO MATCHING SEND
55	ERNOR	NO OUTSTANDING RECEIVE
56	ERIOP	ILLEGAL ORIGIN PORT
57	ERIDP	ILLEGAL DESTINATION PORT
60	ERSEN	INVALID SHARED LIBRARY REFERENCE
61	ERIRL	ILLEGAL RECORD LENGTH SPECIFIED(=0)
62	ERARC	ATTEMPT TO RELEASE CONSOLE DEVICE
63	ERDAI	DEVICE ALREADY IN USE
64	ERARU	ATTEMPT TO RELEASE UNASSIGNED DEVICE
65	ERACU	ATTEMPT TO CLOSE UNOPEN CHANNEL/DEVICE
66	ERITC	I/O TERMINATED BY CLOSE
67	ERLTL	LINE TOO LONG
70	ERPAR	PARITY ERROR
71	EREXC	RESIDENT PROC TRIED TO PUSH (.EXEC)
72	ERNDR	NOT A DIRECTORY
73	ERNSA	SHARED I/O REQUEST NOT TO SHARED AREA
74	ERSNM	ATTEMPT TO CREATE > MAX # SONS
75	ERFIL	FILE READ ERROR
76	ERDTO	DEVICE TIMEOUT
77	ERIOT	WRONG TYPE I/O FOR OPEN TYPE
100	ERFTL	FILENAME TOO LONG
101	ERBOF	POSITIONING BEFORE BEGINNING OF FILE
102	ERPRV	CALLER NOT PRIVILEGED FOR THIS ACTION
103	ERSIM	SIMULTANEOUS REQUESTS ON SAME CHANNEL
104	ERIFT	ILLEGAL FILE TYPE
105	ERNRD	INSUFFICIENT ROOM IN DIRECTORY
106	ERILO	ILLEGAL OPEN
107	ERPRH	ATTEMPT TO ACCESS PROC NOT IN HIERARCHY
110	ERBLR	ATTEMPT TO BLOCK UNBLOCKABLE PROC
111	ERPRE	INVALID SYSTEM CALL PARAMETER
112	ERGES	ATTEMPT TO START MULTIPLE AGENTS
113	ERCIU	CHANNEL IN USE
114	ERICB	INSUFFICIENT CONTIGUOUS DISK BLOCKS
115	ERSTO	STACK OVERFLOW
116	ERIBM	INCONSISTENT BIT MAP DATA
117	ERBSZ	ILLEGAL BLOCK SIZE FOR DEVICE
120	ERXMZ	ATTEMPT TO XMT ILLEGAL MESSAGE

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
121	ERPUF	PHYSICAL UNIT FAILURE
122	ERPWL	PHYSICAL WRITE LOCK
123	ERUOL	PHYSICAL UNIT OFFLINE
124	ERIOO	ILLEGAL OPEN OPTION FOR FILE TYPE
125	ERNDV	TOO MANY OR TOO FEW DEVICE NAMES
126	ERMIS	DISK AND FILE SYS REV #'S DON'T MATCH
127	ERIDD	INCONSISTENT DIB DATA
130	ERILD	INCONSISTENT LD
131	ERIDU	INCOMPLETE LD
132	ERIDT	ILLEGAL DEVICE NAME TYPE
133	ERPDF	ERROR IN PROCESS UST DEFINITION
134	ERVIU	LD IN USE, CANNOT RELEASE
135	ERSRE	SEARCH LIST RESOLUTION ERROR
136	ERCGF	CAN'T GET IPC DATA FROM FATHER
137	ERILB	ILLEGAL LIBRARY NUMBER GIVEN
140	ERRFM	ILLEGAL RECORD FORMAT
141	ERARG	TOO MANY OR TOO FEW ARGUMENTS TO PMGR
142	ERIGM	ILLEGAL ?GTMES PARAMETERS
143	ERICL	ILLEGAL CLI MESSAGE
144	ERMRD	MESSAGE RECEIVE DISABLED
145	ERNAC	NOT A CONSOLE DEVICE
146	ERMIL	ATTEMPT TO EXCEED MAX INDEX LEVEL
147	ERICN	ILLEGAL CHANNEL
150	ERNRR	NO RECEIVER WAITING
151	ERSRR	SHORT RECEIVE REQUEST
152	ERTIN	TRANSMITTER INOPERATIVE
153	ERUNM	ILLEGAL USER NAME
154	ERILN	ILLEGAL LINK #
155	ERDPE	DISK POSITIONING ERROR
156	ERTXT	MSG TEXT LONGER THAN SPEC'D.
157	ERSTR	SHORT TRANSMISSION
160	ERHIS	ERROR ON HISTOGRAM INIT/DELETE
161	ERIRV	ILLEGAL RETRY VALUE
162	ERASS	ASSIGN ERROR - ALREADY YOUR DEVICE
163	ERPET	MAG TAPE REQ PAST LOGICAL END OF TAPE
164	ERSTS	STACK TOO SMALL (?TASK)
165	ERTMT	TOO MANY TASKS REQUESTED (?TASK)
166	ERSOC	SPOOLER OPEN RETRY COUNT EXCEEDED
167	ERACL	ILLEGAL ACL
170	ERWPB	?STMAP BUFFER INVALID OR WRITE PROTECTED

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
171	ERINP	IPC FILE NOT OPENED BY ANOTHER PROC
172	ERFPU	FPU HARDWARE NOT INSTALLED
173	ERPNM	ILLEGAL PROCESS NAME
174	ERPNU	PROCESS NAME ALREADY IN USE
175	ERDCT	DISCONNECT ERROR (MODEM CONTROLLED)
176	ERIPR	NONBLOCKING PROC REQUEST ERROR
177	ERSNI	SYSTEM NOT INSTALLED
200	ERLVL	MAX DIRECTORY TREE DEPTH EXCEEDED
201	ERROO	RELEASING OUT-OF-USE OVERLAY
202	ERRDL	RESOURCE DEADLOCK
203	EREO1	FILE IS OPEN, CAN'T EXCLUSIVE OPEN
204	EREO2	FILE IS EXCLUSIVE OPENED, CAN'T OPEN
205	ERIPD	INIT PRIVILEGE DENIED
206	ERMIM	MULTIPLE ?IMSG CALLS TO SAME DCT
207	ERLNK	ILLEGAL LINK
210	ERIDF	ILLEGAL DUMP FORMAT
211	ERXNA	EXEC NOT AVAILABLE (MOUNT, ETC.)
212	ERXUF	EXEC REQUEST FUNCTION UNKNOWN
213	ERESO	ONLY EXEC'S SONS CAN DO THAT
214	ERRBO	REFUSED BY OPERATOR
215	ERWMT	VOLUME NOT MOUNTED
216	ERISV	ILLEGAL SWITCH VALUE (>65K DECIMAL)
217	ERIFN	INPUT FILE DOES NOT EXIST
220	EROFN	OUTPUT FILE DOES NOT EXIST
221	ERLFN	LIST FILE DOES NOT EXIST
222	ERDFN	DATA FILE DOES NOT EXIST
223	ERGFE	RECURSIVE GENERIC FILE OPEN FAILURE
224	ERNMW	NO MESSAGE WAITING
225	ERNUD	USER DATA AREA DOES NOT EXIST
226	ERDVC	ILLEGAL DEVICE TYPE FROM AOSGEN
227	ERRST	AOS RESTART OF SYSTEM CALL
230	ERFUR	PROBABLY FATAL HARDWARE RUNTIME ERROR
231	ERCFT	USER COMMERCIAL STACK FAULT
232	ERFFT	USER FLOATING POINT STACK FAULT
233	ERUAE	USER DATA AREA ALREADY EXISTS
234	ERISO	ILLEGAL SCREEN_EDIT REQUEST (PMGR)
235	ERDDH	"?SEND" DESTINATION DEVICE HELD BY "^S"
236	EROVR	DATA OVERRUN ERROR
237	ERCPD	CONTROL POINT DIRECTORY MAX SIZE EXCEEDED
240	ERNSD	SYS OR BOOT DISK NOT PART OF MASTER LD

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
241	ERUSY	UNIVERSAL SYSTEM, YOU CAN'T DO THAT
242	EREAD	EXECUTE ACCESS DENIED
243	ERFIX	CAN'T INIT LD, RUN FIXUP ON IT
244	ERFAD	FILE ACCESS DENIED
245	ERDAD	DIRECTORY ACCESS DENIED
246	ERIAD	ATTEMPT TO DEFINE > 1 SPECIAL PROC
247	ERIND	NO SPECIAL PROCESS IS DEFINED
250	ERPRO	ATTEMPT TO ISSUE MCA REQUEST WITH DIRECT I/O IN PROGRESS
251	ERDIO	ATTEMPT TO ISSUE MCA DIRECT I/O WITH OUTSTANDING REQUESTS
252	ERLTK	LAST TASK WAS KILLED
253	ERLRF	RESOURCE LOAD OR RELEASE FAILURE
254	ERNNL	ZERO LENGTH FILENAME SPECIFIED
255	ERBOV	BUFFER OVERFLOW
256	ERNAK	TRANSMISSION FAILURE (NAK COUNT)
257	ERTOF	TRANSMISSION FAILURE (TIMEOUTS)
260	ERDIS	DISCONNECT OCCURRED ON SWITCHED LINE
261	EREOT	EOT CHARACTER RECEIVED
262	EROTH	POSSIBLE LOST DATA ON HASP LINE
263	ERDCU	DCU INOPERATIVE (CAN'T BE INITIALIZED)
264	ERCNV	CONVERSATIONAL REPLY RECEIVED
265	EREPL	END OF POLLING LIST REACHED
266	ERIRT	ILLEGAL RELATIVE TERMINAL NUMBER
267	ERRVI	RVI RESPONSE RECEIVED
270	ERLIN	ILLEGAL LINE NUMBER
271	ERPLS	NOT ENOUGH SPACE FOR POLL LISTS
272	ERCTN	CONTENTION SITUATION WHILE BIDDING
273	ERSEQ	OUT-OF-SEQUENCE GEN ENTRY DURING SINIT
274	ERNSL	ATTEMPT TO ENABLE NON-SYNC LINE
275	ERIMM	NOT ENOUGH MEMORY FOR POLL/SELECT LIST
276	EREPE	LINE ALREADY ENABLED ON ?SEBL CALL
277	ERDSL	LINE ALREADY DISABLED ON ?SDBL CALL
300	ERLNA	I/O REQUEST FOR DISABLED LINE
301	ERLIS	LINE IN SESSION ON ?SSND INITIAL CALL
302	ERSCS	?SSND CONTINUE WITHOUT LINE IN SESSION
303	ERBCT	SEND BYTE COUNT EXCEEDS SYSTEM BUFFER
304	ERBNK	BID ERROR (TOO MANY NAKS)
305	ERWAB	WABT RECEIVED (HASP LINE ONLY)
306	ERBPE	USER BUFFER BYTE POINTER INVALID
307	ERBRT	RETRY COUNT EXCEEDED
310	ERETX	'ETX' CODE RECEIVED

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
311	ERISE	INPUT STATUS ERROR (FORMAT)
312	ERFCT	FAILURE TO CONNECT
313	ERUNI	UNINTERPRETABLE RESPONSE RECEIVED
314	ERENQ	ENQ RECEIVED AFTER TIME-OUT
315	ERCRC	CRC CHECK
316	ERINE	INITIALIZATION PARAMETER ERROR
317	ERTRF	TRANSMITTER FAILURE ERROR
320	ERLNM	LINE NOT MULTIPOINT
321	ERNCS	NOT A CONTROL STATION
322	ERNPL	POLLING LIST NOT DEFINED
323	ERITF	INCOMPATIBLE LPB TAB FORMAT
324	ERPRM	CANNOT DELETE PERMANENT FILE
325	ERSCA	SYSTEM CALL ABORT
326	ERCAD	EXTENDED CONTEXT ALREADY DEFINED
327	ERLAB	UNREADABLE TAPE LABEL
330	ERVOL	INCORRECT LABELED TAPE VOLUME MOUNTED
331	ERFSI	INCORRECT LABELED TAPE FILE SET
332	ERSEC	INCORRECT LABELED TAPE FILE SECTION NUMBER
333	ERGEN	INCORRECT LABELED TAPE FILE GENERATION NUMBER
334	ERVER	INCORRECT LABELED TAPE FILE VERSION NUMBER
335	ERNOA	NO OPERATOR AVAILABLE
336	ERREV	UNKNOWN LABELED TAPE LABEL REVISION
337	ERCAI	EXTENDED CONTEXT ALREADY INITIALIZED
340	ERCNI	EXTENDED CONTEXT NOT INITIALIZED
341	ERCND	EXTENDED CONTEXT NOT DEFINED
342	ERMRL	MEMORY RELEASE ERROR
343	ERITP	TRANSLATION (?READ/?WRITE) ERROR
344	ERNAG	NO SUCH ARGUMENT - ?GTMES
345	ERNCF	NOT IN CLI FORMAT - ?GTMES
346	ERBIF	ILLEGAL BIAS FACTOR
347	ERTLM	CPU TIME LIMIT EXCEEDED
350	ERSMX	ERROR IN SETTING MAX CPU LIMIT
351	ERNM4	ELEMENT SIZE NOT A MULTIPLE OF 4
352	ERWAK	WACK RESPONSE RECEIVED (SYNC LINE)
353	ERNAS	PROCESS IS NOT A SERVER
354	ERCDE	CONNECTION DOESN'T EXIST
355	ERCTF	CONNECTION TABLE FULL
356	ERDIU	DIRECTORY IN USE- CANNOT DELETE
357	ERSHG	ATTEMPT TO GROW AGENT SHARED I/O FILE
360	ERNIN	ILLEGAL DIRECTORY SPECIFICATION

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
361	ERNNA	NETWORK NOT AVAILABLE
362	ERHAE	HOST ALREADY EXIST
363	ERHID	ILLEGAL HOST SPECIFICATION
364	ERHNE	HOST DOES NOT EXIST
365	ERCAH	CAN'T RENAME HOST
366	EREMB	EMPTY MAILBOX ON ?RECNW
367	ERRRR	REMOTE RESOURCE REFERENCE MADE
370	ERCMH	ATTEMPT TO CREATE MULTIPLE LOCAL HOST
371	ERNAI	NOT AWAITING ?IWKUP
372	ERIRP	ILLEGAL REMOTE ?PROC PARAMETER(S)
373	ERIHN	ILLEGAL HOST NAME
374	ERNFC	NOT PROPER FOR A VIRTUAL CIRCUIT
375	ERWSZ	HDLC - INVALID WINDOW SIZE
376	ERFSZ	INVALID FRAME SIZE
377	ERSDA	SEND ACTIVE
400	ERCTY	INVALID CALL TYPE
401	ERDSC	REMOTE IS DISCONNECTING
402	ERRIE	LOCAL RECEIVED INVALID RESPONSE
403	ERRCE	LOCAL RECEIVED CMDR
404	ERCSE	LOCAL IS IN "CAN'T SEND" CONDITION
405	ERLDC	LOCAL IS DISCONNECTING
406	ERRES	LOCAL WAS RESET
407	ERBFO	BUFFER OVERFLOW
410	ERRCA	RECEIVE ACTIVE
411	ERINF	INITIALIZATION FAILED
412	ERINC	LOCAL RECEIVED INVALID COMMAND
413	ERNHL	NON-HDLC ENABLE ATTEMPTED
414	ERKAD	INTERRUPT WAIT TASK ALREADY DEFINED
415	ERDSE	MAP SLOT ERROR
416	ERGBE	GET BUFFER ERROR
417	ERDIE	SYNC DCU INOPERATIVE
420	ERFOE	ERROR OPENING SLDCU.PR
421	ERFRE	ERROR READING SLDCU.PR
422	ERFCE	ERROR CLOSING SLDCU.PR
423	ERGME	ERROR GETTING MEMORY
424	ERUNK	UNKNOWN ERROR
425	ERCBK	CONNECTION HAS BEEN BROKEN
426	ERNDC	ATTEMPT HDLC CALL WITH NO DCU200
427	ERCCS	CANNOT CONNECT TO SELF
430	ERVNC	NO CONNECTION

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
431	ERCDN	CONTROLLER DOES NOT SUPPORT THIS DENSITY MODE.
432	ERITD	INDECIPHERABLE TAPE DENSITY.
433	ERFTM	FILE/TAPE MISMATCH.
434	ERILL	ILLEGAL LABELLED TAPE LEVEL
435	ERILC	ILLEGAL LABELLED TAPE CHARACTER
436	ERISQ	INCORRECT LABELLED TAPE SEQUENCE NUMBER
437	ERIBC	INCORRECT LABELLED TAPE BLOCK COUNT
440	ERVTL	VOLID TOO LONG OR NULL
441	EROTL	OWNER ID TOO LONG
442	ERUTL	USER LABELS TOO LONG OR TOO MANY
443	ERREB	RECORD LENGTH EXCEEDS BLOCK LENGTH
444	ERAPB	AP ALREADY BUSY
445	ERAPN	AP DOES NOT EXIST
446	ERAPL	AP WCS IS NOT LOADED
447	ERAPR	ATTEMPT TO RELEASE A NON-AP PAGE
450	ERAPP	ATTEMPT TO RELEASE AN AP PAGE
451	EROOF	ONLY ONE FILE ON ANSI1 TAPE
452	ERCDU	CANNOT DELETE UNEXPIRED FILE
453	ERNPC	PROCESS CONSOLE DOES NOT EXIST
454	ERHTT	HISTOGRAM TARGET PROCESS TERMINATION
455	ERHIP	ERROR DETECTED WHILE HISTOGRAM IN PROGRESS
456	ERIIP	INVALID IPC MESSAGE
457	ERMUS	MULTIPLE USERS OF FILE, CAN'T TRUNCATE
460	ERSHR	SHARED FILE, CAN'T TRUNCATE
461	ERTRC	FILE BEING TRUNCATED; CANNOT OPEN
462	ERFRM	FRAMING ERROR
463	ERIDI	INTERNAL DIRECTORY INCONSISTENCY
464	ERCOF	COMMERCIAL FAULT
465	ERFXF	FIXED POINT FAULT
466	ERFLF	FLOATING POINT FAULT
467	ER9TP	INCLUDED FOR COMPATIBILITY WITH AOS
470	ERTMR	00470 .DUSR ERTMR= ER9TP+1
471	ERNDS	NO STATISTICS AVAILABLE FOR THIS DEVICE
472	ERNUS	NO STATISTICS AVAILABLE FOR THIS UNIT
473	ERSEX	EP SPACE EXCEEDED
474	ERETL	EP TOO LARGE
475	EREVI	VERSION INCOMPATIBILITY
476	EREDI	EDITREAD STATUS ERROR
477	ERXSP	ILLEGAL EXEC STRING PARAMETER
500	ERXII	BAD INTERNAL EXEC IPC

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
501	ERDUD	DEVICE CODE IS UNDEFINED
502	ERTLK	NO MATCHING ?TLOCK FOR THIS ?TUNLOCK
503	ERNEK	NOT ENOUGH OF THIS TYPE DISK IDENTIFIED TO VSGEN
504	EROUC	TASK HAS AN OUTSTANDING ?UIDCALL
505	ERNUC	TASK HAS NOT BEEN ?UIDCALL'ED
506	EROLF	FILE HAS BAD FCB ADDRESS, NOT AVAILABLE UNTIL FIXUP HAS BEEN RUN ON THIS LD
507	ERRFX	LD WAS RELEASED, REQUIRES FIXUP
510	ERBTC	BAD TRANSFER COUNT FROM CONTROLLER
511	ERIBR	INVALID BAUD RATE FOR THIS DEVICE
512	ERIOM	ILLEGAL OPERATION FOR THIS MEDIA
513	ERDNS	DEVICE NOT SUPPORTED
514	ERNBM	NOT A BITMAP DEVICE
515	ERCMC	CONTROLLER MICROCODE NEEDS UPDATING
516	ERILF	ILLEGAL LABEL FORMAT
517	ERBDK	BAD DISKETTE
520	ERDAE	OPERATOR ALREADY EXISTS
521	ERDNE	TASK IS NOT AN OPERATOR
522	ERNRQ	NO MATCHING OPERATOR REQUEST
523	ERRVN	RESERVED VALUE NOT ZERO
524	ERPVS	PACKET VERSION NOT SUPPORTED
525	ERCRR	CAN'T RENAME THE ROOT
526	ERCDR	CAN'T DELETE THE ROOT
527	ERIOD	ILLEGAL OPEN OPTION FOR THIS DISK
530	ERMNS	MODIFIED SECTOR I/O NOT SUPPORTED BY THIS DEVICE
531	ERCPO	MODIFIED SECTOR I/O NOT SPECIFIED IN OPEN
532	ERCPP	MODIFIED SECTOR I/O REQUIRES PHYSICAL I/O
533	ERINB	INVALID NUMBER OF BLOCKS FOR MODIFIED SECTOR I/O
534	ERNST	TAPE MOUNTED NOT SCRATCH TAPE
535	ERFXR	DISK RELEASED, FIXUP RECOMMENDED ON IT TO RECLAIM SPACE
536	ERNSF	NOT ALL SPACE FREED, FIXUP RECOMMENDED ON THIS DISK
537	ERXND	OUTPUT NOT HELD BY XOFF - NO DATA TO OUTPUT
540	ERXTC	OUTPUT NOT HELD BY XOFF - OUTPUT HELD BY CTS
541	ERXNH	OUTPUT NOT HELD BY XOFF
542	ERIFD	ILLEGAL FUNCTION FOR DEVICE
543	ERPFD	ERROR IN PROFILE FIELD DESCRIPTOR PACKET
544	ERBTS	DESTINATION FIELD TOO SMALL (READ)
545	ERBTL	SOURCE FIELD TOO LARGE (UPDATE)
546	ERPAN	PROFILE ACCESS HAS NOT BEEN INITIATED
547	ERPAI	PROFILE ACCESS ALREADY INITIATED
550	EREDX	PROFILE ENTRY NON-EXISTENT

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
551	ERDRS	DEVICE RESERVED BY ANOTHER PORT
552	ERPOO	ILLEGAL PIPE OPEN OPTION
553	ERIPS	ILLEGAL PIPE SIZE
554	ERPFL	PIPE IS FULL
555	ERMPQ	TOO MANY REQUESTS ON PIPE
556	ERCPA	C PROCESS TRIED TO PROC AN A
557	ERRIP	ILLEGAL RINGLOAD FROM B OR C TYPE PROCESS
560	ERICH	ILLEGAL CHAIN FROM B OR C TYPE PROCESS
561	ERCCA	ILLEGAL CONNECTION ATTEMPT BETWEEN A & C TYPE PROCESS
562	ERLCK	LOCK ALREADY LOCKED - ?FLOCK
563	ERNLK	LOCK NOT LOCKED - ?FUNLOCK
564	ERLLE	LOCK LIMIT EXCEEDED
565	ERLNG	LOCK NOT GRANTED
566	ERPKT	ILLEGAL PACKET ID
567	ERWNE	WINDOW DOES NOT EXIST
570	ERLDX	LDU DOES NOT EXIST
571	ERMSY	MIRRORED LDU IS NOT SYNCHRONIZED
572	ERLMM	LDU NAME MISMATCH - NOT A VALID LDU MIRROR
573	ERLSZ	LDU SIZE MISMATCH - NOT A VALID LDU MIRROR
574	ERRID	LDU IDS ARE NOT UNIQUE - NOT A VALID LDU MIRROR
575	ERMOP	MIRRORED LDU IS OUT OF PHASE
576	ERSFL	MIRRORED LDU SYNCHRONIZATION FAILED
577	ERNAM	LDU IS NOT MIRRORED
600	ERIML	INCOMPLETE MIRRORED LDU SPECIFIED
601	ERLFM	LDU FORMAT MISMATCH - NOT A VALID LDU MIRROR
602	ERCNM	CONTROLLER DOES NOT SUPPORT LDU MIRRORING
603	ERTMM	CONTROLLER CANNOT SUPPORT AN ADDITIONAL LDU MIRROR
604	ERSIP	CANNOT INIT AN LDU IMAGE THAT WAS PREVIOUSLY BEING SYNCHRONIZED
605	ERLDR	LDU RELEASED DURING SYNCHRONIZATION
70003	ERLDE	Lau does not exist
70004	ERLAE	Lau already exists
70005	ERODE	Object does not exist
70006	EROAR	Object already registered
70007	ERTDE	Type does not exist
70010	ERFPD	Function previously defined
70011	ERFOR	Function code out of range
70012	ERFND	Function not defined
70013	ERILT	Invalid lau type
70014	ERIMR	Invalid message received
70015	ERID1	On registering name/uid, uid entry already exists

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
70016	ERID2	On removing lauid/xaddr, no xaddr entry found
70017	ERID3	On deregistering uid, the uid in the corresponding name entry did not match the object's uid
70020	ERID4	On deregistering uid, no corresponding name found
70021	ERID5	On deregistering uid, no lauid entry located
70022	ERIDC	Inconsistent DCALL error
70023	EROGR	Invalid operation on global resource
70024	ERIPG	Invalid operation on process group
70025	ERNGP	Not a global parameter
70026	EROIU	Object in use
70027	ERONR	Object not registered
70030	ERPGN	Process group name does not exist
70031	ERNFO	Not a file system object
70032	ERNUP	Node unreachable permanent
77000	ERXXX	FIRST CODE FOR AOS/VS
77000	ERVWP	INVALID ADDRESS PASSED AS SYSTEM CALL ARGUMENT
77001	ERVBP	INVALID BYTE POINTER PASSED AS SYS CALL ARGUMENT
77002	ERDPT	DIFFERENT TYPE PROCESS(32/16 BIT) WITHOUT PRIVILEGE
77003	ERRAL	RING ALREADY LOADED
77004	ERRNI	RING NUMBER INVALID
77005	ERRTB	RING TOO BIG
77006	ERWSM	SET WKG SET MIN, NOT PRIVILEGED
77007	ERTNE	PMGR- TRACING NOT ENABLED
77010	ERTAE	PMGR- TRACING ALREADY ENABLED
77011	ERNUF	PMGR- TRACING FILE NOT A USER DATA FILE
77012	ERRNA	PMGR- REQUESTOR NOT TRACING AUTHORIZED
77013	ERPNL	PMGR- PATHNAME LENGTH AREA
77014	ERSNF	SYMBOL NOT FOUND IN .ST FILE
77015	ERSNR	SOURCE NOT RESIDENT ON ?LMAP
77016	ERDNR	DESTINATION NOT RESIDENT ON ?LMAP
77017	ERIBP	BKPT SEEN IN USER PROGRAM WHEN DEBUG NOT INIT'ED
77020	ERBST	BAD SYMBOL TABLE FORMAT SEEN (?GTNA CALL)
77021	ERPDO	PAGE FILE DIRECTORY OVERFLOW
77022	ERMWT	MORE THAN ONE WS TRACE DEFINED ON TARGET
77023	ERHWT	BOTH TRACE AND HISTOGRAM CALLED, OR > 1 TRACE
77024	ERDTC	DIFFERENT TYPE CHAIN
77025	ERWST	NO WS TRACE DEFINED ON THIS TARGET
77026	ERWSS	INVALID WKG SET MAX/MIN
77027	ERWSB	INVALID WORKING SET TRACE BUFFER
77030	ERWSF	WORKING SET NOT SWAPPABLE
77031	ERAWM	TRYING TO WIRE MORE PAGES THAN WS MAX

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
77032	ERTPW	TOO MANY PAGES WIRED
77033	ERACC	ACCESS DENIED ON ?VALAD
77034	ERRNL	RING NOT LOADED
77035	ERTAL	TOO MANY ARGUMENTS ON LCALL
77036	ERXBL	?IXIT FROM BASE LEVEL
77037	ERPPR	PMGR PANIC REQUESTED BY NON-PMGR PROCESS
77040	ERSCI	SYSTEM CALL AT INTERRUPT LEVEL
77041	ERNIP	PMGR -- NOT AN IAC PMGR
77042	ERNID	PMGR -- NOT AN IAC-DRIVEN DEVICE
77043	ERSGO	?SIGNL ALREADY OUTSTANDING
77044	ERUFR	UNKNOWN REQUEST FUNCTION
77045	ERIFS	ILLEGAL FED STRING
77046	ERA1O	ATTEMPT TO 1ST OPEN AN OPEN FILE
77047	ERIFI	INVALID PROTECTED FILE ID
77050	ERAPU	ATTEMPT TO PASS UNHELD ACCESS PRIVILEGES
77051	ERNBK	NO BREAKFILE ENABLED FOR THIS RING
77052	ERCDS	PMGR: MODEM DISCONNECT IN PROGRESS - CAN'T OPEN
77053	ERTNF	TASK IS NOT FAULTING
77054	ERNMT	MAP TARGET DOES NOT EXIST
77055	ERMTE	MAP TARGET (ALREADY) MAPPED ELSEWHERE
77056	ERMSI	MAP SPECIFICATION ILLEGAL FOR TARGET
77057	ERRAU	MAP REGION ALREADY IN USE
77060	ERJAI	JP ALREADY INITIALIZED
77061	ERJNI	JP NOT INITIALIZED
77062	ERLNE	LP DOES NOT EXIST
77063	ERLAI	LP ALREADY EXISTS
77064	ERLJP	ATTEMPT TO RELEASE LAST JP ATTACHED TO AN LP
77065	ERIJP	INVALID JPID
77066	ERILP	INVALID LPID
77067	ERJST	JP RUNNING ONE OR MORE SYSTEM TASKS
77070	ERJAA	JP ALREADY ATTACHED TO LP
77071	ERJNA	JP NOT ATTACHED TO LP
77072	ERMLP	ATTEMPT TO EXCEED MAXIMUM LP COUNT
77073	ERJPA	CANNOT DELETE LP WITH JP ATTACHED
77074	ERITI	INVALID TIME INTERVAL
77075	ERICI	INVALID CLASS ID
77076	ERCPC	INVALID CLASS PERCENTAGE
77077	ERIHL	INVALID HIERARCHICAL LEVEL
77100	ERCLU	CLASS IN USE
77101	ERIMP	ILLEGAL BIT MAP

(Continued)

Table A-1. Error Messages in Order of Octal Code

Octal Code	Error Code	Error Message
77102	ERILV	ILLEGAL LOCALITY VALUE
77103	ERLP0	CANNOT DELETE LP 0
77104	ERNMP	NOT A MULTI-PROCESSOR SYSTEM
77105	ERCNE	CLASS DOES NOT EXIST
77106	ERHLP	ILLEGAL HIERARCHY LEVEL / PERCENTAGE PAIR
77107	ERICD	ILLEGAL FUNCTION CODE
77110	ERJPS	JP IS IN A BAD STATE
77111	ERCMM	MICROCODE IS INCOMPATIBLE WITH CURRENT SYSTEM
77112	ERMCR	INCORRECT MICROCODE REVISION
77113	ERMFF	MICROCODE FILE FORMAT ERROR
77114	ERUCP	INVALID CPU MODEL NUMBER
77115	ERCSO	CLASS SCHEDULING IS ENABLED
77116	ERHLT	NON-SEQUENTIAL HIERARCHY LEVELS DESIGNATED
77117	ERPOR	PID IS OUT OF RANGE FOR THIS PROCESS
77120	ERPNO	PROCESS NOT AN OPERATOR
77121	ERBCE	MAX BLOCK COUNT EXCEEDED
77122	ERDEB	DAEMON ERROR IN ERROR BUFFER
77123	ERDRF	DAEMON RESOURCE FAILURE
77124	ERLAS	LOG ALREADY STARTED
77125	ERCL0	CANNOT DELETE CLASS 0
77126	ERPVM	UNKNOWN PRIVILEGE MODE
77127	ERPVX	PRIVILEGE HELD EXCLUSIVELY BY OTHER PROCESS
77130	ERPVO	PRIVILEGE CANNOT BE HELD EXCLUSIVELY
77131	ERPVP	OTHER PROCESSES USING PRIVILEGE
77132	ERWCP	WORKING SET CHANGE ONLY PARTLY DONE
77133	ERFRD	FAULT RECURSION DEPTH EXCEEDED
77134	ERALP	PROCESS'S CLASS NOT SCHEDULABLE ON AN ACTIVE LP
77135	ERCLL	INVALID CELL COUNT
77136	ERNML	NO MICROCODE LOADED IN THIS JP
77137	EREGN	END OF GET NEXT SEQUENCE
77140	ERCTD	CORRUPTED TASK CONTROL BLOCK DATA DETECTED
77141	ERIWR	INVALID WINDOW REFERENCE.
77142	ERWNN	MAXIMUM NUMBER OF WINDOWS EXCEEDED.
77143	ERWMD	WINDOW MARKED FOR DELETION.
77144	ERIGP	INVALID GRAPHICS PARAMETER.
77145	ERIPP	INVALID POINTER DEVICE PARAMETER.
77146	ERIVS	INVALID VIEW OR SCAN PORT.
77147	ERIWO	INVALID WINDOWING OPERATION.
77150	ERIWP	INVALID WINDOWING PARAMETER.

(Concluded)

Licensed Material - Property of Data General Corporation

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ER9TP	467	INCLUDED FOR COMPATIBILITY WITH AOS
ERA1O	77046	ATTEMPT TO 1ST OPEN AN OPEN FILE
ERABT	17	TASK NOT FOUND FOR ABORT
ERACC	77033	ACCESS DENIED ON ?VALAD
ERACL	167	ILLEGAL ACL
ERACU	65	ATTEMPT TO CLOSE UNOPEN CHANNEL/DEVICE
ERADR	6	ILLEGAL STARTING ADDRESS
ERALP	77134	PROCESS'S CLASS NOT SCHEDULABLE ON AN ACTIVE LP
ERAPB	444	AP ALREADY BUSY
ERAPL	446	AP WCS IS NOT LOADED
ERAPN	445	AP DOES NOT EXIST
ERAPP	450	ATTEMPT TO RELEASE AN AP PAGE
ERAPR	447	ATTEMPT TO RELEASE A NON-AP PAGE
ERAPU	77050	ATTEMPT TO PASS UNHELD ACCESS PRIVILEGES
ERARC	62	ATTEMPT TO RELEASE CONSOLE DEVICE
ERARG	141	TOO MANY OR TOO FEW ARGUMENTS TO PMGR
ERARU	64	ATTEMPT TO RELEASE UNASSIGNED DEVICE
ERASS	162	ASSIGN ERROR - ALREADY YOUR DEVICE
ERAWD	34	APPEND AND/OR WRITE ACCESS DENIED
ERAWM	77031	TRYING TO WIRE MORE PAGES THAN WS MAX
ERBCE	77121	MAX BLOCK COUNT EXCEEDED
ERBCT	303	SEND BYTE COUNT EXCEEDS SYSTEM BUFFER
ERBDK	517	BAD DISKETTE
ERBFO	407	BUFFER OVERFLOW
ERBIF	346	ILLEGAL BIAS FACTOR
ERBLR	110	ATTEMPT TO BLOCK UNBLOCKABLE PROC
ERBMX	40	ILLEGAL MAX SIZE ON PROCESS CREATE
ERBNK	304	BID ERROR (TOO MANY NAKS)
ERBOF	101	POSITIONING BEFORE BEGINNING OF FILE
ERBOV	255	BUFFER OVERFLOW
ERBPE	306	USER BUFFER BYTE POINTER INVALID
ERBRT	307	RETRY COUNT EXCEEDED
ERBST	77020	BAD SYMBOL TABLE FORMAT SEEN (?GTNA CALL)
ERBSZ	117	ILLEGAL BLOCK SIZE FOR DEVICE
ERBTC	510	BAD TRANSFER COUNT FROM CONTROLLER
ERBTL	545	SOURCE FIELD TOO LARGE (UPDATE)
ERBTS	544	DESTINATION FIELD TOO SMALL (READ)
ERCAD	326	EXTENDED CONTEXT ALREADY DEFINED
ERCAH	365	CAN'T RENAME HOST
ERCAI	337	EXTENDED CONTEXT ALREADY INITIALIZED

(Continues)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERCBK	425	CONNECTION HAS BEEN BROKEN
ERCCA	561	ILLEGAL CONNECTION ATTEMPT BETWEEN A & C TYPE PROCESS
ERCCS	427	CANNOT CONNECT TO SELF
ERCDE	354	CONNECTION DOESN'T EXIST
ERCDN	431	CONTROLLER DOES NOT SUPPORT THIS DENSITY MODE.
ERCDR	526	CAN'T DELETE THE ROOT
ERCDS	77052	PMGR: MODEM DISCONNECT IN PROGRESS - CAN'T OPEN
ERCDU	452	CANNOT DELETE UNEXPIRED FILE
ERCFT	231	USER COMMERCIAL STACK FAULT
ERCGF	136	CAN'T GET IPC DATA FROM FATHER
ERCIU	113	CHANNEL IN USE
ERCL0	77125	CANNOT DELETE CLASS 0
ERCLL	77135	INVALID CELL COUNT
ERCLU	77100	CLASS IN USE
ERCMC	515	CONTROLLER MICROCODE NEEDS UPDATING
ERCMH	370	ATTEMPT TO CREATE MULTIPLE LOCAL HOST
ERCMM	77111	MICROCODE IS INCOMPATIBLE WITH CURRENT SYSTEM
ERCND	341	EXTENDED CONTEXT NOT DEFINED
ERCNE	77105	CLASS DOES NOT EXIST
ERCNI	340	EXTENDED CONTEXT NOT INITIALIZED
ERCNM	602	CONTROLLER DOES NOT SUPPORT LDU MIRRORING
ERCNV	264	CONVERSATIONAL REPLY RECEIVED
ERCOF	464	COMMERCIAL FAULT
ERCON	42	CONSOLE DEVICE SPECIFICATION ERROR
ERCPA	556	C PROCESS TRIED TO PROC AN A
ERCPC	77076	INVALID CLASS PERCENTAGE
ERCPD	237	CONTROL POINT DIRECTORY MAX SIZE EXCEEDED
ERCPO	531	MODIFIED SECTOR I/O NOT SPECIFIED IN OPEN
ERCPP	532	MODIFIED SECTOR I/O REQUIRES PHYSICAL I/O
ERCRC	315	CRC CHECK
ERCRR	525	CAN'T RENAME THE ROOT
ERCSE	404	LOCAL IS IN "CAN'T SEND" CONDITION
ERCSO	77115	CLASS SCHEDULING IS ENABLED
ERCTD	77140	CORRUPTED TASK CONTROL BLOCK DATA DETECTED
ERCTF	355	CONNECTION TABLE FULL
ERCTN	272	CONTENTION SITUATION WHILE BIDDING
ERCTY	400	INVALID CALL TYPE
ERDAD	245	DIRECTORY ACCESS DENIED
ERDAE	520	OPERATOR ALREADY EXISTS
ERDAI	63	DEVICE ALREADY IN USE

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERDCH	15	DATA CHANNEL MAP FULL
ERDCT	175	DISCONNECT ERROR (MODEM CONTROLLED)
ERDCU	263	DCU INOPERATIVE (CAN'T BE INITIALIZED)
ERDDE	23	DIRECTORY DOES NOT EXIST
ERDDH	235	"?SEND" DESTINATION DEVICE HELD BY "^S"
ERDEB	77122	DAEMON ERROR IN ERROR BUFFER
ERDFN	222	DATA FILE DOES NOT EXIST
ERDID	31	DIRECTORY DELETE ERROR
ERDIE	417	SYNC DCU INOPERATIVE
ERDIO	251	ATTEMPT TO ISSUE MCA DIRECT I/O WITH OUTSTANDING REQUESTS
ERDIS	260	DISCONNECT OCCURRED ON SWITCHED LINE
ERDIU	356	DIRECTORY IN USE- CANNOT DELETE
ERDNE	521	TASK IS NOT AN OPERATOR
ERDNM	45	ILLEGAL DEVICE CODE
ERDNR	77016	DESTINATION NOT RESIDENT ON ?LMAP
ERDNS	513	DEVICE NOT SUPPORTED
ERDPE	155	DISK POSITIONING ERROR
ERDPT	77002	DIFFERENT TYPE PROCESS(32/16 BIT) WITHOUT PRIVILEGE
ERDRF	77123	DAEMON RESOURCE FAILURE
ERDRS	551	DEVICE RESERVED BY ANOTHER PORT
ERDSC	401	REMOTE IS DISCONNECTING
ERDSE	415	MAP SLOT ERROR
ERDSL	277	LINE ALREADY DISABLED ON ?SDBL CALL
ERDTC	77024	DIFFERENT TYPE CHAIN
ERDTO	76	DEVICE TIMEOUT
ERDUD	501	DEVICE CODE IS UNDEFINED
ERDVC	226	ILLEGAL DEVICE TYPE FROM AOSGEN
EREAD	242	EXECUTE ACCESS DENIED
EREDI	476	EDITREAD STATUS ERROR
EREDX	550	PROFILE ENTRY NON-EXISTENT
EREGN	77137	END OF GET NEXT SEQUENCE
EREMB	366	EMPTY MAILBOX ON ?RECNW
ERENQ	314	ENQ RECEIVED AFTER TIME-OUT
ERE01	203	FILE IS OPEN, CAN'T EXCLUSIVE OPEN
ERE02	204	FILE IS EXCLUSIVE OPENED, CAN'T OPEN
EREOF	30	END OF FILE
EREOT	261	EOT CHARACTER RECEIVED
EREPE	276	LINE ALREADY ENABLED ON ?SEBL CALL
EREPL	265	END OF POLLING LIST REACHED
ERESO	213	ONLY EXEC'S SONS CAN DO THAT

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERETL	474	EP TOO LARGE
ERETX	310	'ETX' CODE RECEIVED
EREVI	475	VERSION INCOMPATIBILITY
EREXC	71	RESIDENT PROC TRIED TO PUSH (.EXEC)
ERFAD	244	FILE ACCESS DENIED
ERFCE	422	ERROR CLOSING SLDCU.PR
ERFCT	312	FAILURE TO CONNECT
ERFDE	25	FILE DOES NOT EXIST
ERFFT	232	USER FLOATING POINT STACK FAULT
ERFIL	75	FILE READ ERROR
ERFIX	243	CAN'T INIT LD, RUN FIXUP ON IT
ERFLF	466	FLOATING POINT FAULT
ERFND	70012	Function not defined
ERFNO	2	CHANNEL NOT OPEN
ERFOE	420	ERROR OPENING SLDCU.PR
ERFOR	70011	Function code out of range
ERFPD	70010	Function previously defined
ERFPU	172	FPU HARDWARE NOT INSTALLED
ERFRD	77133	FAULT RECURSION DEPTH EXCEEDED
ERFRE	421	ERROR READING SLDCU.PR
ERFRM	462	FRAMING ERROR
ERFSI	331	INCORRECT LABELED TAPE FILE SET
ERFSZ	376	INVALID FRAME SIZE
ERFTL	100	FILENAME TOO LONG
ERFTM	433	FILE/TAPE MISMATCH.
ERFUR	230	PROBABLY FATAL HARDWARE RUNTIME ERROR
ERFXF	465	FIXED POINT FAULT
ERFXR	535	DISK RELEASED, FIXUP RECOMMENDED ON IT TO RECLAIM SPACE
ERGBE	416	GET BUFFER ERROR
ERGEN	333	INCORRECT LABELED TAPE FILE GENERATION NUMBER
ERGES	112	ATTEMPT TO START MULTIPLE AGENTS
ERGFE	223	RECURSIVE GENERIC FILE OPEN FAILURE
ERGME	423	ERROR GETTING MEMORY
ERGSG	50	ILLEGAL AGENT GATE CALL
ERHAE	362	HOST ALREADY EXIST
ERHID	363	ILLEGAL HOST SPECIFICATION
ERHIP	455	ERROR DETECTED WHILE HISTOGRAM IN PROGRESS
ERHIS	160	ERROR ON HISTOGRAM INIT/DELETE
ERHLP	77106	ILLEGAL HIERARCHY LEVEL / PERCENTAGE PAIR
ERHLT	77116	NON-SEQUENTIAL HIERARCHY LEVELS DESIGNATED

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERHNE	364	HOST DOES NOT EXIST
ERHTT	454	HISTOGRAM TARGET PROCESS TERMINATION
ERHWT	77023	BOTH TRACE AND HISTOGRAM CALLED, OR > 1 TRACE
ERIAD	246	ATTEMPT TO DEFINE > 1 SPECIAL PROC
ERIBC	437	INCORRECT LABELLED TAPE BLOCK COUNT
ERIBM	116	INCONSISTENT BIT MAP DATA
ERIBP	77017	BKPT SEEN IN USER PROGRAM WHEN DEBUG NOT INIT'ED
ERIBR	511	INVALID BAUD RATE FOR THIS DEVICE
ERIBS	44	DEVICE ALREADY IN SYSTEM
ERICB	114	INSUFFICIENT CONTIGUOUS DISK BLOCKS
ERICD	77107	ILLEGAL FUNCTION CODE
ERICH	560	ILLEGAL CHAIN FROM B OR C TYPE PROCESS
ERICI	77075	INVALID CLASS ID
ERICL	143	ILLEGAL CLI MESSAGE
ERICM	1	ILLEGAL SYSTEM COMMAND
ERICN	147	ILLEGAL CHANNEL
ERID1	70015	On registering name/uid, uid entry already exists
ERID2	70016	On removing lauid/xaddr, no xaddr entry found
ERID3	70017	On deregistering uid, the uid in the corresponding name entry did not match the object's uid
ERID4	70020	On deregistering uid, no corresponding name found
ERID5	70021	On deregistering uid, no lauid entry located
ERIDC	70022	Inconsistent DCALL error
ERIDD	127	INCONSISTENT DIB DATA
ERIDF	210	ILLEGAL DUMP FORMAT
ERIDI	463	INTERNAL DIRECTORY INCONSISTENCY
ERIDP	57	ILLEGAL DESTINATION PORT
ERIDT	132	ILLEGAL DEVICE NAME TYPE
ERIDU	131	INCOMPLETE LD
ERIFC	24	ILLEGAL FILENAME CHARACTER
ERIFD	542	ILLEGAL FUNCTION FOR DEVICE
ERIFI	77047	INVALID PROTECTED FILE ID
ERIFN	217	INPUT FILE DOES NOT EXIST
ERIFS	77045	ILLEGAL FED STRING
ERIFT	104	ILLEGAL FILE TYPE
ERIGM	142	ILLEGAL ?GTMES PARAMETERS
ERIGP	77144	INVALID GRAPHICS PARAMETER.
ERIHL	77077	INVALID HIERARCHICAL LEVEL
ERIHN	373	ILLEGAL HOST NAME
ERIIP	456	INVALID IPC MESSAGE
ERIJP	77065	INVALID JPID

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERILB	137	ILLEGAL LIBRARY NUMBER GIVEN
ERILC	435	ILLEGAL LABELLED TAPE CHARACTER
ERILD	130	INCONSISTENT LD
ERILF	516	ILLEGAL LABEL FORMAT
ERILL	434	ILLEGAL LABELLED TAPE LEVEL
ERILN	154	ILLEGAL LINK #
ERILO	106	ILLEGAL OPEN
ERILP	77066	INVALID LPID
ERILT	70013	Invalid lau type
ERILV	77102	ILLEGAL LOCALITY VALUE
ERIML	600	INCOMPLETE MIRRORED LDU SPECIFIED
ERIMM	275	NOT ENOUGH MEMORY FOR POLL/SELECT LIST
ERIMP	77101	ILLEGAL BIT MAP
ERIMR	70014	Invalid message received
ERINB	533	INVALID NUMBER OF BLOCKS FOR MODIFIED SECTOR I/O
ERINC	412	LOCAL RECEIVED INVALID COMMAND
ERIND	247	NO SPECIAL PROCESS IS DEFINED
ERINE	316	INITIALIZATION PARAMETER ERROR
ERINF	411	INITIALIZATION FAILED
ERINP	171	IPC FILE NOT OPENED BY ANOTHER PROC
ERIOD	527	ILLEGAL OPEN OPTION FOR THIS DISK
ERIOM	512	ILLEGAL OPERATION FOR THIS MEDIA
ERIOO	124	ILLEGAL OPEN OPTION FOR FILE TYPE
ERIOP	56	ILLEGAL ORIGIN PORT
ERIOT	77	WRONG TYPE I/O FOR OPEN TYPE
ERIPD	205	INIT PRIVILEGE DENIED
ERIPG	70024	Invalid operation on process group
ERIPP	77145	INVALID POINTER DEVICE PARAMETER.
ERIPR	176	NONBLOCKING PROC REQUEST ERROR
ERIPS	553	ILLEGAL PIPE SIZE
ERIRB	20	INSUFFICIENT ROOM IN BUFFER
ERIRL	61	ILLEGAL RECORD LENGTH SPECIFIED(=0)
ERIRP	372	ILLEGAL REMOTE ?PROC PARAMETER(S)
ERIRT	266	ILLEGAL RELATIVE TERMINAL NUMBER
ERIRV	161	ILLEGAL RETRY VALUE
ERISE	311	INPUT STATUS ERROR (FORMAT)
ERISO	234	ILLEGAL SCREEN_EDIT REQUEST (PMGR)
ERISQ	436	INCORRECT LABELLED TAPE SEQUENCE NUMBER
ERISV	216	ILLEGAL SWITCH VALUE (>65K DECIMAL)
ERITC	66	I/O TERMINATED BY CLOSE

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERITD	432	INDECIPHERABLE TAPE DENSITY.
ERITF	323	INCOMPATIBLE LPB TAB FORMAT
ERITI	77074	INVALID TIME INTERVAL
ERITP	343	TRANSLATION (?READ/?WRITE) ERROR
ERIVP	53	INVALID PORT NUMBER
ERIVS	77146	INVALID VIEW OR SCAN PORT.
ERIWO	77147	INVALID WINDOWING OPERATION.
ERIWP	77150	INVALID WINDOWING PARAMETER.
ERIWR	77141	INVALID WINDOW REFERENCE.
ERJAA	77070	JP ALREADY ATTACHED TO LP
ERJAI	77060	JP ALREADY INITIALIZED
ERJNA	77071	JP NOT ATTACHED TO LP
ERJNI	77061	JP NOT INITIALIZED
ERJPA	77073	CANNOT DELETE LP WITH JP ATTACHED
ERJPS	77110	JP IS IN A BAD STATE
ERJST	77067	JP RUNNING ONE OR MORE SYSTEM TASKS
ERKAD	414	INTERRUPT WAIT TASK ALREADY DEFINED
ERLAB	327	UNREADABLE TAPE LABEL
ERLAE	70004	Lau already exists
ERLAI	77063	LP ALREADY EXISTS
ERLAS	77124	LOG ALREADY STARTED
ERLCK	562	LOCK ALREADY LOCKED - ?FLOCK
ERLDC	405	LOCAL IS DISCONNECTING
ERLDE	70003	Lau does not exist
ERLDR	605	LDU RELEASED DURING SYNCHRONIZATION
ERLDX	570	LDU DOES NOT EXIST
ERLFM	601	LDU FORMAT MISMATCH - NOT A VALID LDU MIRROR
ERLFN	221	LIST FILE DOES NOT EXIST
ERLIN	270	ILLEGAL LINE NUMBER
ERLIS	301	LINE IN SESSION ON ?SSND INITIAL CALL
ERLJP	77064	ATTEMPT TO RELEASE LAST JP ATTACHED TO AN LP
ERLLE	564	LOCK LIMIT EXCEEDED
ERLMM	572	LDU NAME MISMATCH - NOT A VALID LDU MIRROR
ERLNA	300	I/O REQUEST FOR DISABLED LINE
ERLNE	77062	LP DOES NOT EXIST
ERLNG	565	LOCK NOT GRANTED
ERLNK	207	ILLEGAL LINK
ERLNM	320	LINE NOT MULTIPOINT
ERLP0	77103	CANNOT DELETE LP 0
ERLRF	253	RESOURCE LOAD OR RELEASE FAILURE

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERLSZ	573	LDU SIZE MISMATCH - NOT A VALID LDU MIRROR
ERLTK	252	LAST TASK WAS KILLED
ERLTL	67	LINE TOO LONG
ERLVL	200	MAX DIRECTORY TREE DEPTH EXCEEDED
ERMCR	77112	INCORRECT MICROCODE REVISION
ERMEM	5	INSUFFICIENT MEMORY AVAILABLE
ERMFF	77113	MICROCODE FILE FORMAT ERROR
ERMIL	146	ATTEMPT TO EXCEED MAX INDEX LEVEL
ERMIM	206	MULTIPLE ?IMSG CALLS TO SAME DCT
ERMIS	126	DISK AND FILE SYS REV #'S DON'T MATCH
ERMLP	77072	ATTEMPT TO EXCEED MAXIMUM LP COUNT
ERMNS	530	MODIFIED SECTOR I/O NOT SUPPORTED BY THIS DEVICE
ERMOP	575	MIRRORED LDU IS OUT OF PHASE
ERMPQ	555	TOO MANY REQUESTS ON PIPE
ERMPR	16	SYSTEM CALL PARAMETER ADDRESS ERROR
ERMRD	144	MESSAGE RECEIVE DISABLED
ERMRL	342	MEMORY RELEASE ERROR
ERMSI	77056	MAP SPECIFICATION ILLEGAL FOR TARGET
ERMSY	571	MIRRORED LDU IS NOT SYNCHRONIZED
ERMTE	77055	MAP TARGET (ALREADY) MAPPED ELSEWHERE
ERMUS	457	MULTIPLE USERS OF FILE, CAN'T TRUNCATE
ERMWT	77022	MORE THAN ONE WS TRACE DEFINED ON TARGET
ERNAC	145	NOT A CONSOLE DEVICE
ERNAD	27	NON-DIRECTORY ARGUMENT IN PATHNAME
ERNAE	26	FILE NAME ALREADY EXISTS
ERNAG	344	NO SUCH ARGUMENT - ?GTMES
ERNAI	371	NOT AWAITING ?IWKUP
ERNAK	256	TRANSMISSION FAILURE (NAK COUNT)
ERNAM	577	LDU IS NOT MIRRORED
ERNAS	353	PROCESS IS NOT A SERVER
ERNBK	77051	NO BREAKFILE ENABLED FOR THIS RING
ERNBM	514	NOT A BITMAP DEVICE
ERNCF	345	NOT IN CLI FORMAT - ?GTMES
ERNCS	321	NOT A CONTROL STATION
ERNDC	426	ATTEMPT HDLC CALL WITH NO DCU200
ERNDR	72	NOT A DIRECTORY
ERNDS	471	NO STATISTICS AVAILABLE FOR THIS DEVICE
ERNDV	125	TOO MANY OR TOO FEW DEVICE NAMES
ERNEF	52	IPC MESSAGE EXCEEDS BUFFER LENGTH
ERNEK	503	NOT ENOUGH OF THIS TYPE DISK IDENTIFIED TO VSGEN

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERNFC	374	NOT PROPER FOR A VIRTUAL CIRCUIT
ERNFO	70031	Not a file system object
ERNGP	70025	Not a global parameter
ERNHL	413	NON-HDLC ENABLE ATTEMPTED
ERNID	77042	PMGR -- NOT AN IAC-DRIVEN DEVICE
ERNIN	360	ILLEGAL DIRECTORY SPECIFICATION
ERNIP	77041	PMGR -- NOT AN IAC PMGR
ERNLK	563	LOCK NOT LOCKED - ?FUNLOCK
ERNM4	351	ELEMENT SIZE NOT A MULTIPLE OF 4
ERNMC	35	NO CHANNELS AVAILABLE
ERNML	77136	NO MICROCODE LOADED IN THIS JP
ERNMP	77104	NOT A MULTI-PROCESSOR SYSTEM
ERNMS	54	NO MATCHING SEND
ERNMT	77054	MAP TARGET DOES NOT EXIST
ERNMW	224	NO MESSAGE WAITING
ERNNA	361	NETWORK NOT AVAILABLE
ERNNL	254	ZERO LENGTH FILENAME SPECIFIED
ERNOA	335	NO OPERATOR AVAILABLE
ERNOR	55	NO OUTSTANDING RECEIVE
ERNOT	11	NO TASK CONTROL BLOCK AVAILABLE
ERNPC	453	PROCESS CONSOLE DOES NOT EXIST
ERNPL	322	POLLING LIST NOT DEFINED
ERNRD	105	INSUFFICIENT ROOM IN DIRECTORY
ERNRQ	522	NO MATCHING OPERATOR REQUEST
ERNRR	150	NO RECEIVER WAITING
ERNSA	73	SHARED I/O REQUEST NOT TO SHARED AREA
ERNSD	240	SYS OR BOOT DISK NOT PART OF MASTER LD
ERNSF	536	NOT ALL SPACE FREED, FIXUP RECOMMENDED ON THIS DISK
ERNSL	274	ATTEMPT TO ENABLE NON-SYNC LINE
ERNST	534	TAPE MOUNTED NOT SCRATCH TAPE
ERNSW	43	SWAP FILE SPACE EXHAUSTED
ERNUC	505	TASK HAS NOT BEEN ?UIDCALL'ED
ERNUD	225	USER DATA AREA DOES NOT EXIST
ERNUF	77011	PMGR- TRACING FILE NOT A USER DATA FILE
ERNUP	70032	Node unreachable permanent
ERNUS	472	NO STATISTICS AVAILABLE FOR THIS UNIT
EROAR	70006	Object already registered
ERODE	70005	Object does not exist
EROFN	220	OUTPUT FILE DOES NOT EXIST
EROGR	70023	Invalid operation on global resource

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
EROIU	70026	Object in use
EROLF	506	FILE HAS BAD FCB ADDRESS, NOT AVAILABLE UNTIL FIXUP HAS BEEN RUN ON THIS LD
ERONR	70027	Object not registered
EROOF	451	ONLY ONE FILE ON ANSI1 TAPE
EROPR	3	CHANNEL ALREADY OPEN
EROTH	262	POSSIBLE LOST DATA ON HASP LINE
EROTL	441	OWNER ID TOO LONG
EROUC	504	TASK HAS AN OUTSTANDING ?UIDCALL
EROVN	7	ILLEGAL OVERLAY NUMBER
EROVR	236	DATA OVERRUN ERROR
ERPAI	547	PROFILE ACCESS ALREADY INITIATED
ERPAN	546	PROFILE ACCESS HAS NOT BEEN INITIATED
ERPAR	70	PARITY ERROR
ERPDF	133	ERROR IN PROCESS UST DEFINITION
ERPDO	77021	PAGE FILE DIRECTORY OVERFLOW
ERPET	163	MAG TAPE REQ PAST LOGICAL END OF TAPE
ERPFD	543	ERROR IN PROFILE FIELD DESCRIPTOR PACKET
ERPFL	554	PIPE IS FULL
ERPGN	70030	Process group name does not exist
ERPKT	566	ILLEGAL PACKET ID
ERPLS	271	NOT ENOUGH SPACE FOR POLL LISTS
ERPNL	77013	PMGR- PATHNAME LENGTH AREA
ERPNM	173	ILLEGAL PROCESS NAME
ERPNO	77120	PROCESS NOT AN OPERATOR
ERPNU	174	PROCESS NAME ALREADY IN USE
ERPOO	552	ILLEGAL PIPE OPEN OPTION
ERPOR	77117	PID IS OUT OF RANGE FOR THIS PROCESS
ERPPR	77037	PMGR PANIC REQUESTED BY NON-PMGR PROCESS
ERPRE	111	INVALID SYSTEM CALL PARAMETER
ERPRH	107	ATTEMPT TO ACCESS PROC NOT IN HIERARCHY
ERPRM	324	CANNOT DELETE PERMANENT FILE
ERPRN	51	NUMBER OF PROCESSES EXCEEDS MAX
ERPRO	250	ATTEMPT TO ISSUE MCA REQUEST WITH DIRECT I/O IN PROGRESS
ERPRP	37	ILLEGAL PRIORITY
ERPRV	102	CALLER NOT PRIVILEGED FOR THIS ACTION
ERPTY	41	ILLEGAL PROCESS TYPE
ERPUF	121	PHYSICAL UNIT FAILURE
ERPVM	77126	UNKNOWN PRIVILEGE MODE
ERPVO	77130	PRIVILEGE CANNOT BE HELD EXCLUSIVELY
ERPVP	77131	OTHER PROCESSES USING PRIVILEGE

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERPVS	524	PACKET VERSION NOT SUPPORTED
ERPVX	77127	PRIVILEGE HELD EXCLUSIVELY BY OTHER PROCESS
ERPWL	122	PHYSICAL WRITE LOCK
ERQTS	13	ERROR IN QTASK REQUEST
ERRAD	33	READ ACCESS DENIED
ERRAL	77003	RING ALREADY LOADED
ERRAU	77057	MAP REGION ALREADY IN USE
ERRBO	214	REFUSED BY OPERATOR
ERRCA	410	RECEIVE ACTIVE
ERRCE	403	LOCAL RECEIVED CMDR
ERRDL	202	RESOURCE DEADLOCK
ERREB	443	RECORD LENGTH EXCEEDS BLOCK LENGTH
ERRES	406	LOCAL WAS RESET
ERREV	336	UNKNOWN LABELED TAPE LABEL REVISION
ERRFM	140	ILLEGAL RECORD FORMAT
ERRFX	507	LD WAS RELEASED, REQUIRES FIXUP
ERRID	574	LDU IDS ARE NOT UNIQUE - NOT A VALID LDU MIRROR
ERRIE	402	LOCAL RECEIVED INVALID RESPONSE
ERRIP	557	ILLEGAL RINGLOAD FROM B OR C TYPE PROCESS
ERRMP	47	ERROR ON REMAP CALL
ERRNA	77012	PMGR- REQUESTOR NOT TRACING AUTHORIZED
ERRNI	77004	RING NUMBER INVALID
ERRNL	77034	RING NOT LOADED
ERROO	201	RELEASING OUT-OF-USE OVERLAY
ERRRR	367	REMOTE RESOURCE REFERENCE MADE
ERRST	227	AOS RESTART OF SYSTEM CALL
ERRTB	77005	RING TOO BIG
ERRVI	267	RVI RESPONSE RECEIVED
ERRVN	523	RESERVED VALUE NOT ZERO
ERSAL	4	SHARED I/O REQ NOT MAP SLOT ALIGNED
ERSCA	325	SYSTEM CALL ABORT
ERSCI	77040	SYSTEM CALL AT INTERRUPT LEVEL
ERSCS	302	?SSND CONTINUE WITHOUT LINE IN SESSION
ERSDA	377	SEND ACTIVE
ERSEC	332	INCORRECT LABELED TAPE FILE SECTION NUMBER
ERSEN	60	INVALID SHARED LIBRARY REFERENCE
ERSEQ	273	OUT-OF-SEQUENCE GEN ENTRY DURING SINIT
ERSEX	473	EP SPACE EXCEEDED
ERSFL	576	MIRRORED LDU SYNCHRONIZATION FAILED
ERSFT	22	USER STACK FAULT

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERSGO	77043	?SIGNL ALREADY OUTSTANDING
ERSHG	357	ATTEMPT TO GROW AGENT SHARED I/O FILE
ERSHP	46	ERROR ON SHARED PARTITION SET
ERSHR	460	SHARED FILE, CAN'T TRUNCATE
ERSIM	103	SIMULTANEOUS REQUESTS ON SAME CHANNEL
ERSIP	604	CANNOT INIT AN LDU IMAGE THAT WAS PREVIOUSLY BEING SYNCHRONIZED
ERSMX	350	ERROR IN SETTING MAX CPU LIMIT
ERSNF	77014	SYMBOL NOT FOUND IN .ST FILE
ERSNI	177	SYSTEM NOT INSTALLED
ERSNM	74	ATTEMPT TO CREATE > MAX # SONS
ERSNR	77015	SOURCE NOT RESIDENT ON ?LMAP
ERSOC	166	SPOOLER OPEN RETRY COUNT EXCEEDED
ERSPC	21	FILE SPACE EXHAUSTED
ERSRE	135	SEARCH LIST RESOLUTION ERROR
ERSRL	36	RELEASE OF NON-ACTIVE SHARED SLOT
ERSRR	151	SHORT RECEIVE REQUEST
ERSTO	115	STACK OVERFLOW
ERSTR	157	SHORT TRANSMISSION
ERSTS	164	STACK TOO SMALL (?TASK)
ERTAE	77010	PMGR- TRACING ALREADY ENABLED
ERTAL	77035	TOO MANY ARGUMENTS ON LCALL
ERTDE	70007	Type does not exist
ERTID	14	TASK I.D. ERROR
ERTIM	10	ILLEGAL TIME ARGUMENT
ERTIN	152	TRANSMITTER INOPERATIVE
ERTLK	502	NO MATCHING ?TLOCK FOR THIS ?TUNLOCK
ERTLM	347	CPU TIME LIMIT EXCEEDED
ERTMM	603	CONTROLLER CANNOT SUPPORT AN ADDITIONAL LDU MIRROR
ERTMR	470	00470 .DUSR ERTMR= ER9TP+1
ERTMT	165	TOO MANY TASKS REQUESTED (?TASK)
ERTNE	77007	PMGR- TRACING NOT ENABLED
ERTNF	77053	TASK IS NOT FAULTING
ERTOF	257	TRANSMISSION FAILURE (TIMEOUTS)
ERTPW	77032	TOO MANY PAGES WIRED
ERTRC	461	FILE BEING TRUNCATED; CANNOT OPEN
ERTRF	317	TRANSMITTER FAILURE ERROR
ERTXT	156	MSG TEXT LONGER THAN SPEC'D.
ERUAE	233	USER DATA AREA ALREADY EXISTS
ERUCP	77114	INVALID CPU MODEL NUMBER
ERUFR	77044	UNKNOWN REQUEST FUNCTION

(Continued)

Table A-2. Error Messages in Alphabetical Order

Error Code	Octal Code	Error Message
ERUNI	313	UNINTERPRETABLE RESPONSE RECEIVED
ERUNK	424	UNKNOWN ERROR
ERUNM	153	ILLEGAL USER NAME
ERUOL	123	PHYSICAL UNIT OFFLINE
ERUSY	241	UNIVERSAL SYSTEM, YOU CAN'T DO THAT
ERUTL	442	USER LABELS TOO LONG OR TOO MANY
ERVBP	77001	INVALID BYTE POINTER PASSED AS SYS CALL ARGUMENT
ERVER	334	INCORRECT LABELED TAPE FILE VERSION NUMBER
ERVIU	134	LD IN USE, CANNOT RELEASE
ERVNC	430	NO CONNECTION
ERVOL	330	INCORRECT LABELED TAPE VOLUME MOUNTED
ERVTL	440	VOLID TOO LONG OR NULL
ERVWP	77000	INVALID ADDRESS PASSED AS SYSTEM CALL ARGUMENT
ERWAB	305	WABT RECEIVED (HASP LINE ONLY)
ERWAD	32	WRITE ACCESS DENIED
ERWAK	352	WACK RESPONSE RECEIVED (SYNC LINE)
ERWCP	77132	WORKING SET CHANGE ONLY PARTLY DONE
ERWMD	77143	WINDOW MARKED FOR DELETION.
ERWMT	215	VOLUME NOT MOUNTED
ERWNE	567	WINDOW DOES NOT EXIST
ERWNN	77142	MAXIMUM NUMBER OF WINDOWS EXCEEDED.
ERWPB	170	?STMAP BUFFER INVALID OR WRITE PROTECTED
ERWSB	77027	INVALID WORKING SET TRACE BUFFER
ERWSF	77030	WORKING SET NOT SWAPPABLE
ERWSM	77006	SET WKG SET MIN, NOT PRIVILEGED
ERWSS	77026	INVALID WKG SET MAX/MIN
ERWST	77025	NO WS TRACE DEFINED ON THIS TARGET
ERWSZ	375	HDLC - INVALID WINDOW SIZE
ERXBL	77036	?IXIT FROM BASE LEVEL
ERXII	500	BAD INTERNAL EXEC IPC
ERXMT	12	SIGNAL TO ADDRESS ALREADY IN USE
ERXMZ	120	ATTEMPT TO XMT ILLEGAL MESSAGE
ERXNA	211	EXEC NOT AVAILABLE (MOUNT, ETC.)
ERXND	537	OUTPUT NOT HELD BY XOFF - NO DATA TO OUTPUT
ERXNH	541	OUTPUT NOT HELD BY XOFF
ERXSP	477	ILLEGAL EXEC STRING PARAMETER
ERXTC	540	OUTPUT NOT HELD BY XOFF - OUTPUT HELD BY CTS
ERXUF	212	EXEC REQUEST FUNCTION UNKNOWN
ERXXX	77000	FIRST CODE FOR AOS/VS

(Concluded)

End of Appendix

B

Sample Programs

This appendix contains 11 program sets that illustrate AOS/VS and AOS/DVS system calls. We have written 10 of the program sets in assembly language and the eleventh in FORTRAN 77.

Table B–1 lists the names of the programs in each program set along with the major system concepts and calls that the program set illustrates. Complete explanations of each program set, with program listings, appear after Table B–1.

The source program files, except for the FORTRAN 77 ones, are part of each release of AOS/VS and of AOS/DVS. We encourage you to create the program (.PR) files and obtain results equivalent to those in this appendix. We also encourage you to experiment with the programs by changing them and observing the effects of the changes. If your system has FORTRAN 77 software you can key in the relatively small FORTRAN 77 source program files.

Table B-1. Sample Program Sets and their Descriptions

Program Set Number and Program Names	Summary Description
1. HEAR.SR, SPEAK.SR, SON.SR (subroutine)	These source program units produce two executable programs that illustrate interprocess communication. The system calls are ?PROC, ?ILKUP, ?ISEND, and ?IREC.
2. RUNTIME.SR	This program unit obtains its process's runtime statistics via system call ?RUNTM.
3. RINGLOAD.SR, INRING.SR, GATE.ARRAY.SR (subroutine)	These source program units produce two executable programs. RINGLOAD.PR loads INRING.PR into an inner ring and executes it -- illustrating instruction LCALL, how to define a gate array, and system call ?RINGLD.
4. FILCREATE.SR	It opens the terminal and creates a disk file, showing I/O calls ?OPEN and ?CREATE.
5. WRITE.SR	It opens the terminal and a disk file, gets text from the terminal, writes it to the disk file, and plays back the text on demand. The related system calls are ?OPEN, ?READ, ?WRITE, and -- to show file positioning -- ?SPOS.
6. DLIST.SR	It lists all filenames in a directory to the line printer. It shows call ?GTMES plus I/O via calls ?OPEN, ?GOPEN, ?READ/?WRITE, ?GNFN, and ?SEND.
7. NEWTASK.SR	This program shows a simple use of multitasking. It issues system calls ?TASK, ?MYTID, and ?IDKIL.
8. BOOMER.SR	Here is a fast multitasked file copy program that uses two synchronized tasks to read and write data.
9. TIMEOUT.SR	This program delays itself for a specified period. It illustrates system calls ?WDELAY and ?GTMES.
10. DIRCREATE.F77, CHECK.F77 (subroutine)	This program is a FORTRAN 77 equivalent of the sample assembly language program in Chapter 1. DIRCREATE.PR issues ?CREATE (directory version) and ?DIR system calls.
11. CREATE_ WINDOW.SR	This program uses calls ?WINDOW and ?PROC as it creates and deletes a window.

Licensed Material - Property of Data General Corporation

Program Set 1 — HEAR.SR, SPEAK.SR, SON.SR

SON.SR is a subroutine that issues ?PROC to create and start a process. HEAR.SR/SON.SR and SPEAK.SR form two complete programs — HEAR.PR and SPEAK.PR — that illustrate interprocess communications. Program HEAR uses subroutine SON to execute program SPEAK. HEAR then issues an ?IREC call for a message from SPEAK, and SPEAK sends the message via an ?ISEND call. Each program uses ?ILKUP to discover the other's port number.

Figure B-1 contains HEAR.SR, Figure B-2 contains SPEAK.SR, and Figure B-3 contains SON.SR.

```
; This program module, HEAR.SR, calls module
;    SON to create and run program SPEAK.PR.
;    SPEAK then tries to send an IPC message to HEAR.
;    HEAR's destination IPC port is PORTR; its
;    origin port is PORTS.  After program SPEAK
;    sends the message, it terminates; then
;    program HEAR passes SPEAK's  message to the
;    CLI on termination.  The Link lines are
;
;       X  LINK HEAR SON     and
;       X  LINK SPEAK
;
; To use routine SON, you need "Create without
;    Block", "Use IPC", and "SONs" privileges.
        .TITLE  HEAR
        .EXTN   SON          ; External address.
        .ENT    HEAR         ; Begin here.
        .NREL                ; Default to partition 4.
        ; Open terminal for input and output.
HEAR:   ?OPEN   CONSOLE      ; For Input and Output.
        WBR     E.RROR       ; Error, process it.
        ?WRITE  CONSOLE      ; Write message (byte pointer is
                             ;    already in I/O packet.)
        WBR     E.RROR       ; Error, process it.
        ; Start the SON process to run SPEAK.PR.
        XLEFB   0, SPEAK*2 ; Get byte pointer to filename.
        XJSR    SON          ; SON creates process.
        ; SPEAK is running.  Create IPC entry for receive.
        XLEFB   0,PORTR*2  ; Byte pointer to port name.
        ?CREATE  IPCEN       ; Create IPC entry PORTR.
        WBR     E.RROR       ; Error, process it.
        XLEFB   0, MES1*2  ; Byte pointer to text message.
        XWSTA   0, CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE      ; Write message.
        WBR     E.RROR       ; Error, process it.
```

Figure B-1. Listing of Program HEAR.SR (Continues)

```
        ; See if SPEAK's entry, PORTS, has been created.
GETOP:  XLEFB   0, PORTS*2 ; Byte pointer to port name.
        ?ILKUP             ; Get port number in AC1.
        WBR     TEST       ; Error -- check for
                           ;    ERDNE (doesn't exist).
        XLEFB   0, MES2*2  ; Entry exists, get byte pointer.
        XWSTA   0, CONSOLE+?IBAD  ; Put in I/O packet.
        ?WRITE  CONSOLE    ; Write "Success" message.
        WBR     E.RROR     ; Error, process it.

        XWSTA   1, RHDR+?IOPH  ; Put origin port number in
                           ;        receive header (Note
                           ;        Wide Store instruction.)
        NLDAI   1, 0       ; Generate 1.
        XNSTA   0, RHDR+?IDPN  ; Put destination port 1 in
                           ;        receive header (Note
                           ;        Narrow Store instruction.)
        ?IREC   RHDR       ; Receive SPEAK message.
        WBR     E.RROR     ; Error, process it.
        XLEFB   1, MSBUF*2 ; Message received; get byte pointer
                           ;    to message buffer.
        NLDAI   ?RFCF+100., 2  ; Put flag + 100. words for
                           ;        message in AC2.
        ?RETURN            ; Return with message to CLI.
        WBR     E.RROR     ; Return error.

        ; ?ILKUP error. Check for "does not exist" error
        ;     and delay if this was the error.
TEST:   NLDAI   ERFDE, 1   ; Put ERFDE number in AC1.
        WSEQ    0, 1       ; Skip if ERFDE.
        WBR     E.RROR     ; Handle other error.
        XLEFB   0, MES3*2  ; ERFDE. Get message byte pointer.
        XWSTA   0, CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE    ; Write message.
        WBR     E.RROR     ; Error, process it.

        NLDAI   5000., 0   ; 5000 milliseconds = 5 seconds.
        ?WDELAY            ; Delay for 5 seconds.
        WBR     E.RROR     ; Error, process it.
        WBR     GETOP      ; Try ?ILKUP again.

        ; Error instructions, pointer, file/port names.
E.RROR: XJMP    ERROR      ; To error handler.
;.SON:  SON                ; To subroutine SON.

SPEAK:  .TXT  "SPEAK.PR"   ; SPEAK program filename.
PORTR:  .TXT  "PORTR"      ; Receive port name.
PORTS:  .TXT  "PORTS"      ; Send port name.
```

Figure B-1. Listing of Program HEAR.SR (Continued)

Licensed Material - Property of Data General Corporation

```
        ; Set up open and I/O packet for terminal.
        .ENABLE WORD            ; Most entries = 16 bits.
                                ; Use WORD for rest of program.
CONSOLE: .BLK   ?IBLT           ; Packet length.

        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO       ; Data-sensitive records,
                                ;   input and output.

        .LOC    CONSOLE+?IMRS
         -1     ; Memory block size, default to 2048.

        .LOC    CONSOLE+?IBAD
        .DWORD  MES*2           ; 2-word byte pointer to text/buffer.

        .LOC    CONSOLE+?IRCL
        120.    ; Maximum record length of 120 characters.

        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2           ; 2-word byte pointer to filename.

        .LOC    CONSOLE+?IDEL
        .DWORD  -1              ; Data-sensitive delimiter table address;
                                ;    default to NL, CR, NUL.

        .LOC    CONSOLE+?IBLT ; Default others to 0.
        ; End of console I/O packet.
; Filename, buffer, messages. .NOLOC 1 follows ...
        .NOLOC  1               ;  ... to save listing space.
CON:    .TXT "@CONSOLE"         ; Use generic name.
MES:    .TXT "From HEAR -- have opened terminal; ready to call SON.<12>"
MES1:   .TXT "From HEAR -- back from SON, SPEAK is running. Created
 IPC entry.<12>"
MES2:   .TXT "From HEAR -- have ?ILKUPed the IPC port entry.<12>"
MES3:   .TXT "From HEAR -- Error on ?ILKUP. Will delay and
 try again.<12>"
        .NOLOC  0               ; Resume listing all.
```

Figure B-1. Listing of Program HEAR.SR (Continued)

```
; Header for IPC entry.
IPCEN:  .LOC    IPCEN+?CFTYP

        ?FIPC                   ; FIPC type file.
        .LOC    IPCEN+?CPOR
        1                       ; Port number is 1.

        .LOC    IPCEN+?CTIM
        .DWORD  -1              ; Default to current time.

        .LOC    IPCEN+?CACP
        .DWORD  -1              ; Default to current ACL.
                                ; Receive header RHDR -- for the ?IREC.
RHDR:   .LOC    RHDR+?ISFL
        0                       ; No system flags.

        .LOC    RHDR+?IUFL
        0                       ; No user flags.

        .LOC    RHDR+?IOPH
        .DWORD  0               ; For origin port number.

        .LOC    RHDR+?IDPN
        0                       ; For destination port number.

        .LOC    RHDR+?ILTH
        100.                    ; Message buffer, 100. words.

        .LOC    RHDR+?IPTR
        .DWORD  MSBUF           ; Message buffer address.

        .LOC    RHDR+?PLTH ; End of header.
MSBUF:  .BLK    101.            ; Message buffer.
        ;  Error handler.
ERROR:  NLDAI   ?RFEC+?RFCF+?RFER, 2  ; Error flags.
        ?RETURN                 ; To CLI.
        WBR     ERROR           ; Return error.

        .END    HEAR
```

Figure B–1. Listing of Program HEAR.SR (Concluded)

```
        .TITLE  SPEAK
        .ENT    SPEAK
        .NREL
; The sole purpose of this program is to send
;    an IPC message to a brother process, then to
;    terminate itself. SPEAK's origin port is
;    PORTS and its destination port is PORTR.
        ; Create an IPC entry port named PORTS.
SPEAK:  XLEFB   0, PORTS*2 ; Byte pointer to port name.
        ?CREATE IPCEN      ; Create IPC port.
        WBR     E.RROR     ; Error, process it.
        ; See if receive entry, PORTR, has been created.
GETNM:  XLEFB   0, PORTR*2 ; Byte pointer to port name.
        ?ILKUP             ; Put port number in AC1.
        WBR     TEST       ; Error -- check for
                           ; ERDNE (doesn't exist).
        XWSTA   1, SHDR+?IDPH  ; Entry exists; put
                               ;    number in send header.
        NLDAI   1, 0           ; Generate 1.
        XNSTA   0, SHDR+?IOPN  ; Put destination port 1
                               ;    in send header. Note
                               ;    narrow store instruction.
        ?ISEND  SHDR           ; Send SPEAK message.
        WBR     E.RROR         ; Error, process it.
        ; Message sent. Delay to allow brother to receive
        ;    message before terminating yourself.
        NLDAI   10000.,  0     ; 10000 ms = 10 seconds.
        ?WDELAY                ; Delay for 10 seconds.
        WBR     E.RROR         ; Error, process it.
        NLDAI   -1, 0          ; Get -1 to terminate self.
        WSUB    2,2            ; No IPC message to father.
        ?TERM                  ; The end.
        WBR     E.RROR         ; Termination error, handle it.
        ; ?ILKUP error. Check for "does not exist" error and delay
        ;    if this was the error.
TEST:   NLDAI   ERFDE,  1      ; Put ERFDE number in AC1.
        WSEQ    0, 1           ; Skip if ERFDE.
        WBR     E.RROR         ; Handle other error.
        NLDAI   5000.,  0      ; 5000 milliseconds =  5 seconds.
        ?WDELAY                ; Delay for 5 seconds.
        WBR     E.RROR         ; Error, process it.
        WBR     GETNM          ; Try ?ILKUP again.
        ; Error instruction, pointer, file/port names.
E.RROR: XJMP    ERROR          ; To error handler.
PORTS:  .TXT    "PORTS"        ; Send port name.
PORTR:  .TXT    "PORTR"        ; Receive port name.
```

Figure B–2. Listing of Program SPEAK.SR (Continues)

```
        ; Header for IPC entry.
        .ENABLE WORD                    ; Most entries = 16 bits.
                                        ;    Use WORD for rest of prog.
IPCEN:  .LOC    IPCEN+?CFTYP
        ?FIPC                           ; FIPC type file.

        .LOC    IPCEN+?CPOR
        1                               ; Port number is 1.

        .LOC    IPCEN+?CTIM
        .DWORD  -1                      ; Default to current time.

        .LOC    IPCEN+?CACP
        .DWORD  -1                      ; Default to current ACL.
        ; Send header SHDR -- for the ?ISEND.
SHDR:   .LOC    SHDR+?ISFL
        0                               ; No system flags.

        .LOC    SHDR+?IUFL
        0                               ; No user flags.

        .LOC    SHDR+?IDPH
        .DWORD  0                       ; For destination port number.

        .LOC    SHDR+?IOPN
        0                               ; For origin port number.

        .LOC    SHDR+?ILTH
        100.                            ; Message buffer, 100. words.

        .LOC    SHDR+?IPTR
        .DWORD  MSBUF                   ; Message buffer address.

        .LOC    SHDR+?PLTH              ; End of header.
        ; Message to send. A .NOLOC 1 follows ...
        .NOLOC  1                       ; ... to save listing space.
        MSBUF:  .TXT  "Hello, this is your son SPEAKing. As you <12>
read these words, I am terminating and so are you.<12>"
        .NOLOC  0                       ; Resume listing all.
        ;  Error handler.
ERROR:  NLDAI   ?RFEC+?RFCF+?RFER,  2   ; Error flags.
        ?RETURN                         ; To CLI.
        WBR     ERROR                   ; Return error.
        .END    SPEAK
```

Figure B-2. Listing of Program SPEAK.SR (Concluded)

```
        .TITLE  SON
        .ENT    SON
        .NREL

; This subroutine, SON.SR, creates a swappable
;   son process. The process runs program SPEAK.PR,
;   but you can change this by replacing SPEAK.PR
;   with any executable-program-name.PR.  All
;   defaultable ?PROC characteristics are
;   defaulted: No IPC header, processname is its
;   PID, same maximum size as the calling process,
;   same generic files and username and privileges
;   as the father, no sons. To use SON, you must
;   have "Create without Block" privilege in your
;   user profile.

;  Get program name to ?PROC.

SON:    WSSVR   0               ; Save return from XJSR.
        XLEFB   0, PRGNM*2      ; Byte pointer to program name.
        XWSTA   0,PKT+?PSNM     ; Put in ?PROC packet.
        ?PROC   PKT             ; Create process.
        WBR     ERROR           ; Error, handle it.
        WRTN                    ; To caller.

PRGNM:  .TXT    "SPEAK.PR"

ERROR:  NLDAI   ?RFEC+?RFCF+?RFER,2  ; Error flags.
        ?RETURN                 ; To CLI.
        WBR     ERROR           ; Return error.

        ; ?PROC packet.

        .ENABLE WORD            ; Most entries = 16 bits.

       PKT:
        .BLK    ?PLTH           ; Packet length.

        .LOC    PKT+?PFLG
        0                       ;  Default creation specifications.

        .LOC    PKT+?PPRI
        -1                      ; Priority (default = caller's).

        .LOC    PKT+?PSNM
        .DWORD  PRGNM*2         ; 2-word byte pointer to process name.

        .LOC    PKT+?PIPC
        .DWORD  -1              ; 2-word address of IPC message header;
                                ;    there is no header.

        .LOC    PKT+?PNM
        .DWORD  -1              ; 2-word byte pointer to process name;
                                ;    default = PID.

        .LOC    PKT+?PMEM
        .DWORD  -1              ; Maximum memory pages;
                                ;    default = caller's.

        .LOC    PKT+?PDIR
        .DWORD  -1              ; 2-word byte pointer to directory;
                                ;    default = working directory.

        .LOC    PKT+?PCON
        .DWORD  0               ; 2-word byte pointer to @CON name. None.
```

Figure B-3. Listing of Program SON.SR (Continues)

```
.LOC    PKT+?PCAL
-1                          ; Default = 2 concurrent system calls.

.LOC    PKT+?PWSS
-1                          ; Maximum working set; default=caller's.

.LOC    PKT+?PUNM
.DWORD  -1                  ; Username; default = caller's.

.LOC    PKT+?PPRV
?PVIP                       ; Privileges; allow ?ISEND call.

.LOC    PKT+?PPCR
0                           ; Number of processes son can create.

.LOC    PKT+?PWMI
-1                          ; Minimum working set; default=caller's.

.LOC    PKT+?PIFP
.DWORD  0                   ; 2-word byte pointer to @INPUT name;
                            ;    none.

.LOC    PKT+?POFP
.DWORD  0                   ; 2-word byte pointer to @OUTPUT name;
                            ;    none.

.LOC    PKT+?PLFP
.DWORD  0                   ; 2-word byte pointer to @LISTFILE name;
                            ;    none.

.LOC    PKT+?PDFP
.DWORD  0                   ; 2-word byte pointer to @DATAFILE name;
                            ;    none.

.LOC    PKT+?SMCH
.DWORD  -1                  ; Maximum CPU time; default to father's.

.LOC    PKT+?PLTH
; End of ?PROC packet.
.END
```

Figure B-3. Listing of Program SON.SR (Concluded)

Licensed Material - Property of Data General Corporation

The assembly commands are

```
XEQ MASM HEAR
XEQ MASM SON
XEQ MASM SPEAK
```

The Link commands are

```
XEQ LINK HEAR SON
XEQ LINK SPEAK
```

The command to execute both programs is

```
XEQ HEAR
```

You need "USE IPC" and "CREATE WITHOUT BLOCK" privileges in your user profile to execute the programs.

The console output from the execution command is next.

```
From HEAR -- have opened terminal; ready to call SON.
From HEAR -- back from SON, SPEAK is running. Created IPC entry.
From HEAR -- Error on ?ILKUP. Will delay and try again.
From HEAR -- have ?ILKUPed the IPC port entry.

Hello, this is your son SPEAKing. As you
read these words, I am terminating and so are you.
```

Program Set 2 — RUNTIME.SR

Program RUNTIME uses system call ?RUNTM to get two of its own runtime statistics. They are the process's CPU time in milliseconds (offset ?GRCH) and CPU usage in page-seconds (offset ?GRPH). Then, RUNTIME converts them to a string of ASCII decimal digits, displays them, and loops. The program continues this ?RUNTM-display loop until you type the three-character terminating sequence ST<NL>.

Figure B-4 contains RUNTIME.SR.

```
; This routine gets and displays two runtime statistics
;    on a process. It opens the terminal, issues ?RUNTM,
;    converts and displays the statistics. It gets its
;    own statistics, but you can have it get any process's
;    statistics by passing the process's filename.PR or
;    PID. To use it as a subroutine, start with proper Save,
;    and end with proper Return.
        .TITLE  RUNTIME
        .ENT    RUNTIME, CONVERT
        .NREL

        ; Open terminal for I/O.
RUNTIME:?OPEN   CONSOLE ; Open for input and output,
                        ;    with start message.
        WBR     ERROR   ; Error, handle it.
        ?WRITE  CONSOLE ; Write initial message.
        WBR     ERROR   ; Error, handle it.

        ; Call ?RUNTM to get statistics. (For other process'
        ; statistics, put name or PID in AC0 and -1 or 0 in AC1.)
LOOP:   NLDAI   -1, 0     ; Check self.
        ?RUNTM  RPKT      ; Get statistics in packet.
        WBR     ERROR     ; Error, process it.
        XWLDA   1, MSECS  ; Get milliseconds from RPKT.
        XLEFB   2, MSECMSG*2     ; Milliseconds message byte address.
        XJSR    CONVERT   ; Convert milliseconds to ASCII
                          ;    decimal digits, put them in message.
        XLEFB   0,MSECMSG*2      ; Message done, get byte pointer.
        XWSTA   0,CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE   ; Write milliseconds message.
        WBR     ERROR     ; Error, process it.

        XWLDA   1,PSECS   ; Get page-seconds from RPKT.
        XLEFB   2, PSECMSG*2     ; Page-seconds message byte address.
        XJSR    CONVERT   ; Convert value to ASCII
                          ;    decimal digits, put them in message.
        XLEFB   0,PSECMSG*2      ; Message done, get byte pointer.
        XWSTA   0,CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE   ; Write page-seconds message.
        WBR     ERROR     ; Error, process it.
```

Figure B-4. Listing of Program RUNTIME.SR (Continues)

Licensed Material - Property of Data General Corporation

```
        ; See if user wants to stop.
        XLEFB   0, BUF*2 ; Byte pointer to I/O buffer.
        XWSTA   0,CONSOLE+?IBAD ; Put in I/O packet.
        ?READ   CONSOLE  ; Read for terminator.
        WBR     ERROR    ; Process error.
        NLDAI   'ST', 0  ; Put 'ST' in AC0.
        XNLDA   1, BUF   ; Put first word of buffer in AC1.
        WSNE    0,1      ; Skip if first word not 'ST'.
        WBR     BYE      ; Good bye.
        WBR     LOOP     ; Do it all again.

        ;  Error handler and return.
ERROR:  NLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
BYE:    WSUB    2,2              ; Good return flgs.
        ?RETURN                  ; To father.
        WBR     ERROR            ; Return error.

        ; Open and I/O packet for console.

        .ENABLE WORD
        ; Use WORD for rest of program.

CONSOLE: .BLK   ?IBLT            ; Packet length.
        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO        ; Data-sensitive records, Input/Output.
        .LOC    CONSOLE+?IMRS
         -1                      ; Memory block size, default to 2048.
        .LOC    CONSOLE+?IBAD
        .DWORD  ITEXT*2          ; 2-word byte pointer to text/buffer.
        .LOC    CONSOLE+?IRCL
        120.                     ; Maximum record length of 120 chars.
        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2            ; 2-word byte pointer to filename.
        .LOC    CONSOLE+?IDEL
        .DWORD  -1               ; Data-sensitive delimiter table
                                 ;    address; default to NL, CR, NUL.
        .LOC    CONSOLE+?IBLT    ; Default others to 0.

        ; End of console I/O packet.

        ; Filename, start message, buffer.  A .NOLOC 1 follows ...
        .NOLOC  1                ; ... to save listing space.
CON:    .TXT "@CONSOLE"          ; Use generic name.

ITEXT:  .TXT "I give runtime statistics on a process.
 Type ST[NL] to return to father.<212><12>"

BUF:    .BLK (BUF-CON)*2         ; Use number of bytes in message.
        .NOLOC  0                ; Resume listing.

        ; Messages to include converted statistics. .NOLOC here ...
        .NOLOC  1                ; ... to save listing space.
MSECMSG: .TXT "         Milliseconds elapsed.<12>"

PSECMSG: .TXT "         Page-seconds used. Type
 ST [NEWLINE] to stop, other char to loop.<212><12>"
        .NOLOC  0
```

Figure B–4. Listing of Program RUNTIME.SR (Continued)

```
        ; ?RUNTM Packet.
RPKT:   .BLK    ?GRLTH          ; Length of packet.
SECS:   .LOC    RPKT+?GRRH
        .DWORD  0               ; System returns elapsed time in seconds.
MSECS:  .LOC    RPKT+?GRCH
        .DWORD  0               ; System returns CPU time in milliseconds.
IO:     .LOC    RPKT+?GRIH
        .DWORD  0               ; System returns I/O usage.
PSECS:  .LOC    RPKT+?GRPH
        .DWORD  0               ; System returns CPU time in page-seconds.
        .LOC    RPKT+?GRLTH     ; End of packet.
; Convert routine converts binary value to decimal and puts it in text
;    string. Enter with AC1 containing value, AC2 containing byte
;    address of text message.
CONVERT: WSSVS  0               ; Save return.
        WMOV    2,3             ; Use AC3 for byte pointer shifting.
        WADI    3,3             ; Add integer 3 to byte address.
        NLDAI   10., 2          ; Put 10. in AC2.
DLOOP:  WSUB    0,0             ; Zero AC0 (high-order part
                                ;    of dividend). AC1 still has
                                ;    low order part of dividend.
        WDIVS                   ; Divide by 10., put quotient
                                ;    in AC1, remainder in AC0.
        IORI    60,0            ; OR in 60 for ASCII number.
        WSTB    3, 0            ; Store AC0-byte (bits 24-31)
                                ;    in byte address of AC3.
        WSBI    1, 3            ; Decrement the byte address.
        MOV     1,1,SNR         ; Did quotient go to 0?
        WRTN                    ; Yes, done, return to caller.
        WBR     DLOOP           ; No, do another digit.
        .END    RUNTIME
```

Figure B–4. Listing of Program RUNTIME.SR (Concluded)

The assembly command is

XEQ MASM RUNTIME

The Link command is

XEQ LINK RUNTIME

The command to execute the program is

XEQ RUNTIME

A typical line of console output from the execution command is next.

```
 59   Milliseconds elapsed.
  3   Page-seconds used. Type ST [NEW LINE] to stop, other char to loop.
```


Licensed Material – Property of Data General Corporation

Program Set 3 — RINGLOAD.SR, INRING.SR, and GATE.ARRAY.SR

These three source files form two programs that illustrate call ?RINGLD. The first source, RINGLOAD.SR, forms a complete program that executes in ring 7. The second module, INRING, must be linked with the third module, subroutine GATE.ARRAY. Together they form program file INRING.PR. RINGLOAD.PR loads INRING.PR into ring 5, where it executes and passes control to the debugger. After you type ESC-R to continue the program, control returns to RINGLOAD.PR in ring 7.

These programs show use of the pseudo-operation ?EXTG, the instruction LCALL, and the system call ?RINGLD.

Figure B-5 contains RINGLOAD.SR, Figure B-6 contains INRING.SR, and Figure B-7 contains GATE.ARRAY.SR.

```
; This program, RINGLOAD, loads program INRING
;    into an inner ring, then calls INRING via
;    LCALL. The inner ring can be 4, 5, or 6,
;    depending on the Link switch used on INRING.
;    RINGLOAD assumes that INRING.PR, linked from
;    INRING and GATE.ARRAY exists.
        .TITLE  RINGLOAD
        .ENT    RINGLOAD
        .NREL           ; Default unshared code.
; Open terminal, ?RINGLD program.
RINGLOAD: ?OPEN CONSOLE ; Open terminal for I/O.
        WBR     ERROR   ; Error, process it.
        ?WRITE  CONSOLE ; Write "About to.." message.
        WBR     ERROR   ; Error, process it.
        XLEFB   0, PNAME*2 ; Byte pointer to INRING name.
        ?RINGLD         ; Load INRING.
        WBR     ERROR   ; Error, process it.
; Do LCALL.  Real format is
;
;       LCALL displ index arg-count.
;
; Like a system call, LCALL has a single-word error return
;    location and double-word good return location:
;
;       LCALL  displ  index  arg-count
;       error return     (single-word)
;       normal return    (double-word)
        LCALL   INRING,0,0      ; Call it, index, argument count zero.
        WBR     INERROR         ; Report INRING error.
        ; Back from INRING. Depart with message for CLI.
        XLEFB   0, MES2*2       ;  Byte pointer to farewell message.
        XWSTA   0, CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE         ; Write message.
        WBR     ERROR           ; Error, process it.
        WSUB    2, 2            ; Set for good return.
        WBR     BYE             ; Done, give message and depart.
```

Figure B-5. Listing of Program RINGLOAD.SR (Continues)

```
        ; Inner program and current program error handlers.
INERROR: LLEFB  0, INMES*2       ; Get "Inner prog" error message byte ptr.
        LWSTA   0, CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE          ; Write to terminal.
        WBR     ERROR            ; Write error.
ERROR:  WLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
BYE:    ?RETURN                  ; To CLI.
        WBR     ERROR            ; Return error.

        ; Definition of inner program ring bracket and gate.
        ;    The system uses this, not program name (INRING)
        ;    to access gate and inner program.
INRING = 5S3+0                   ; Ring 5 + 1st gate.

        ; Messages. A .NOLOC 1 follows.

        .NOLOC  1

PNAME:  .TXT "INRING.PR"

MES1:   .TXT  "I'm RINGLOAD, about to ?RINGLD
 program INRING.<12>"

MES2:   .TXT "<212>I'm RINGLOAD, back from
 INRING, terminating.<12>"

INMES:  .TXT  "<212>Error in inner ring program.<12>"

        .NOLOC  0    ; Resume listing all.

        ; Open and I/O packet -- needed for I/O.
        .ENABLE WORD ; Most entries =16 bits.
CONSOLE: .BLK   ?IBLT            ; Packet length.
        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO        ; Data-sensitive records, Input/Output.
        .LOC    CONSOLE+?IMRS
         -1                      ; Memory block size, default to 2048.
        .LOC    CONSOLE+?IBAD
        .DWORD  MES1*2           ; 2-word byte pointer to text/buffer.
        .LOC    CONSOLE+?IRCL
        120.                     ; Maximum record length of 120 chars.
        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2            ; 2-word byte pointer to filename.
        .LOC    CONSOLE+?IDEL
        .DWORD  -1               ; Data sensitive delimiter table add; none.
        .LOC    CONSOLE+?IBLT    ; Default others to 0.
        ; End of console I/O packet.
CON:    .TXT  "@CONSOLE"

        .END    RINGLOAD
```

Figure B-5. Listing of Program RINGLOAD.SR (Concluded)

Licensed Material - Property of Data General Corporation

```
; This program, INRING, is loaded by program
;    RINGLOAD, then called via LCALL. This program
;    saves the return address, opens the terminal,
;    writes  messages, invokes the debugger (so
;    that you can look around in inner ring). Then,
;    after you type ESC-R, it returns to RINGLOAD in
;    ring 7. All code is shared except for I/O packet.
;
; This program must be linked with module GATE.ARRAY.
;    It can be executed in ring 4, 5, or 6 depending on
;    three things: the LCALL name definition in the
;    calling program, the gate defined in GATE.ARRAY,
;    and the Link switch used on this program. Here, the
;    caller RINGLOAD defines the name as gate 5 (5S3);
;    GATE.ARRAY defines gate 5; and the Link command
;    used is    X LINK/RING=5 INRING GATE.ARRAY.

        .TITLE  INRING
        .ENT    INRING
        .NREL   1           ; Shared code (SC). Can be up
                            ;    to 1,024,000 bytes.

        ; Save return, open terminal, write message, go into
        ;    the debugger.

INRING: WSAVR   0           ; Save frame (4 ACs, PC in AC3)
        ?OPEN   CONSOLE     ; Open console for I/O.
        WBR     ERTN        ; Error, process it.
        ?WRITE  CONSOLE     ; Write message from ring 5.
        WBR     ERTN        ; Error, process it.
        ?DEBUG              ; Go into the debugger.
        WBR     ERTN        ; Error, process it.
        LLEFB   0, MES2*2 ; Get byte pointer to return message.
        LWSTA   0, CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE     ; Write another message.
        WBR     ERTN        ; Error, process it.

        ; Done. Ready for good return to caller.

        LDAFP   3           ; Frame pointer in AC3.
        XWISZ   0, 3        ; Increment return address for good
                            ;    return to LCALLer.
        WRTN                ; Return to caller.

        ; INRING error handler. Returns error to the outer ring
        ;    caller, not to the CLI.

ERTN:   LDAFP   3           ; Get the frame pointer.
        LWSTA   0, ?OAC0, 3 ; Put error code (AC0)
                            ;       in saved frame's AC0.
                            ; In frame, ?OAC0 is offset of old AC0.
        WRTN                ; Return to LCALLer's error return

        ; Text messages.  .NOLOC 1 follows.
        .NOLOC  1
MES1:   .TXT  "I'm INRING, in inner ring. About
 to ?DEBUG. Type ESC R to proceed.<212><12>"

MES2:   .TXT  "<212><212>From INRING -- about
 to WRTN.<12>"
        .NOLOC  0           ; Resume listing all.

        .NREL               ; Must use unshared for packet because
                            ;    program and system write into it.
```

Figure B-6. Listing of Program INRING.SR (Continues)

```
        ; Open and I/O packet for @CONSOLE.
         .ENABLE WORD      ; Most entries = 16 bits.
CONSOLE: .BLK   ?IBLT      ; Packet length.
        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO ; Data-sensitive records, Input/Output.
        .LOC    CONSOLE+?IMRS
         -1                ; Memory block size, default to 2048.
        .LOC    CONSOLE+?IBAD
        .DWORD  MES1*2     ; 2-word byte pointer to text/buffer.
        .LOC    CONSOLE+?IRCL
        120.               ; Maximum record length of 120 characters.
        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2      ; 2-word byte pointer to filename.
        .LOC    CONSOLE+?IDEL
        .DWORD  -1         ; Data-sensitive delimiter table address; default.
        .LOC    CONSOLE+?IBLT ; Default others to 0.
        ; End of console I/O packet.
        ; Filename. A .NOLOC 1 follows.
        .NOLOC  1
CON:    .TXT   "@CONSOLE"
        .NOLOC  0          ; Resume listing all.
        .END    INRING
```

Figure B-6. Listing of Program INRING.SR (Concluded)

```
; This module defines the gate array. Generally,
;    the gate array must be defined in a separate
;    module. It must contain
;       .EXTG prog-entry-name
;    where prog-entry-name is the start entry name
;    in the program that will be accessed through
;    the gate. And it must be Linked with the inner
;    ring program; here:
;       X LINK/RING=5 INRING GATE.ARRAY
        .TITLE  GATE.ARRAY
        .EXTG   INRING     ; EXTGate defines gate
                           ;    for program INRING.
        ; Define 2-word pointer to gate array.
        .LOC    34         ; Locations 34 and 35.
        .DWORD  GATE.ARRAY  ; Pointer.
        .NREL 1            ; Shared code for general use.
        GATE.ARRAY: .DWORD  1    ; Gate array, 1 gate.
        .DWORD  (RING7-RING5)+INRING ; Address will be determined
                           ;    by Link.  Allows gate to be accessed
                           ;    by a program in any ring.
        RING7 = 7S3        ; Bits 1 3 specify gate 7.
        RING5 = 5S3        ; Bits 1 3 specify gate 5.
        .END
```

Figure B-7. Listing of Program GATE.ARRAY.SR

Licensed Material - Property of Data General Corporation

The assembly commands are

```
XEQ MASM RINGLOAD
XEQ MASM INRING
XEQ MASM GATE.ARRAY
```

The Link commands are

```
XEQ LINK RINGLOAD
XEQ LINK/RING=5 INRING GATE.ARRAY
```

The command to execute both programs is

```
XEQ RINGLOAD
```

The console output from the execution command is next.

```
I'm RINGLOAD, about to ?RINGLD program INRING.
I'm INRING, in inner ring. About to ?DEBUG. Type ESC R to proceed.

DEBUG 4    AOS/VS User Debugger    Revision 006.000
34000001344 00000000000 12000000470 12001776206 00000000000
Warning- Memory Modification not allowed
_$R

From INRING -- about to WRTN.

I'm RINGLOAD, back from INRING, terminating.
```

Program Set 4 — FILCREATE.SR

Program FILCREATE opens the terminal and asks you for the name of a file to create. If the file already exists, FILCREATE deletes and recreates it.

Figure B-8 contains FILCREATE.SR.

```
; This program opens the terminal and asks for a
;    filename. It then creates the file (deleting it
;    first if it exists) with an ACL of OWR. (An IPC
;    ?CREATE example occurs in programs SPEAK and HEAR.)
        .TITLE  FILCREATE
        .ENT    FILCREATE
        .NREL
FILCREATE: ?OPEN CONSOLE ; For input and output
        WBR     ERROR    ; Error, process it.
        ?WRITE  CONSOLE  ; Ask for filename.
        WBR     ERROR    ; Error, process it.
        XLEFB   0,BUF*2  ; Get buffer byte pointer.
        XWSTA   0,CONSOLE+?IBAD ; Put in I/O packet.
        ?READ   CONSOLE  ; Read filename.
        WBR     ERROR    ; Error, process it.
CREATE: ?CREATE CPKT     ; Create file (ACO still has
                         ;    byte pointer to the name).
        WBR     TEST     ; Error, check for ERNAE.
        ?WRITE  CONSOLE  ; Done, confirm name.
        WBR     ERROR    ; Error, process it.
        XLEFB   0, TMES*2 ; Get confirmation message byte pointer.
        XWSTA   0,CONSOLE+?IBAD ; Put it in I/O packet.
        ?WRITE  CONSOLE  ; Write confirmation message.
        WBR     ERROR    ; Error, process it.
        WSUB    2,2      ; Good return flags.
        ?RETURN          ; Good bye.
        WBR     ERROR    ; Return error.
TEST:   WLDAI   ERNAE, 2 ; Code = 'already exists'.
        WSEQ    2,0      ; Skip if error is ERNAE.
        WBR     ERROR    ; Other error, process it.
        XLEFB   0, BUF*2 ; Get buffer byte pointer.
        ?DELETE          ; Delete the file.
        WBR     ERROR    ; Error, process it.
        WBR     CREATE   ; Go and create the file.
ERROR:  WLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
        ?RETURN          ; To CLI.
        WBR     ERROR    ; Return error.
```

Figure B-8. Listing of Program FILCREATE.SR (Continues)

Licensed Material - Property of Data General Corporation

```
        ; Create Packet.
        .ENABLE WORD ; Most entries = 16 bits.
CPKT:   .BLK    ?CLTH     ; Packet length.
        .LOC    CPKT+?CFTYP
        ?ORDS*400+?FUDF   ; Data sensitive records, left byte
                          ;    (format mnemonic must always be *400)
                          ;    + User Data File.
        .LOC    CPKT+?CCPS
        0                 ; Fixed record length; ignored here.
        .LOC    CPKT+?CTIM
        .DWORD  -1        ; Default to current time.
        .LOC    CPKT+?CACP
        .DWORD  ACL*2     ; 2-word byte pointer to ACL.
        .LOC    CPKT+?CDEH
        0                 ; Reserved for system.
        .LOC    CPKT+?CDEL
        -1                ; Default element size.
        .LOC    CPKT+?CMIL
        -1                ; Default number of index levels.
        .LOC    CPKT+?CLTH ; Default others.
        ; End of Create packet.
ACL:    .TXT "DOCUMENT<0><?FACO+?FACW+?FACR><0>"
        ; Use your own username instead of DOCUMENT.
        ; Open and I/O packet -- needed for I/O.
CONSOLE: .BLK   ?IBLT     ; Packet length.
        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO ; Data-sensitive records, Input/Output.
        .LOC    CONSOLE+?IMRS
         -1               ; Memory block size, default to 2048.
        .LOC    CONSOLE+?IBAD
        .DWORD  ITEXT*2   ; 2-word byte pointer to text/buffer.
        .LOC    CONSOLE+?IRCL
        120.              ; Maximum record length of 120 characters.
        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2     ; 2-word byte pointer to filename.
        .LOC    CONSOLE+?IDEL
        .DWORD  -1        ; Data-sensitive delimiter table address;
                          ;    default to NL, CR, NUL.
        .LOC    CONSOLE+?IBLT ; Default others to 0.
        ; End of console I/O packet.
; Filename, buffer, messages. .NOLOC 1 follows.
        .NOLOC  1         ; To save listing space.
CON:    .TXT "@CONSOLE"   ; Use generic name.
ITEXT:  .TXT "Type valid filename to be created.  "
BUF:    .BLK    50.       ; Filename buffer, 100 chars.
TMES:   .TXT " created with ACL of OWR.<12>"
        .NOLOC  0         ; Resume listing all.
        .END    FILCREATE
```

Figure B-8. Listing of Program FILCREATE.SR (Concluded)

The assembly, link, and execute commands are

```
XEQ MASM FILCREATE
XEQ LINK FILCREATE
XEQ FILCREATE
```

Typical lines of console dialog from the execution command are next.

```
) XEQ FILCREATE
Type valid filename to be created.  FOO
FOO
 created with ACL of OWR.


) ACL/V FOO
FOO                 DOCUMENT,OWR
```


Licensed Material - Property of Data General Corporation

Program Set 5 — WRITE.SR

Program WRITE opens the terminal and disk file FILE. It then asks you to type lines on the keyboard, and writes each line as a record into file FILE. When you type **RD<NL>**, program WRITE reads the records back from FILE, one at a time, and writes them on the terminal. WRITE includes two ?OPEN system calls (one with file deletion/creation) plus at least one ?READ, ?WRITE, and ?SPOS (for file positioning).

Figure B-9 contains WRITE.SR.

```
; This program opens the terminal (@CONSOLE), then
;    opens a disk file named FILE. This opening
;    creates FILE, deleting it first if it exists.
;    The program then reads input from the terminal,
;    a line at a time, and writes it to file FILE.
;    When the user types RD, the program positions
;    FILE at its beginning, writes all of its records
;    (lines) to the terminal, and ends. The example
;    shows calls ?OPEN, ?OPEN with file create/deletion,
;    ?READ, ?WRITE, ?SPOS, and ?CLOSE. There are
;    two ?OPEN packets: 1 for @CONSOLE, 1 for FILE.
        .TITLE  WRITE
        .ENT    WRITE
        .NREL   ; Default to partition 4.

        ; Open terminal (@CONSOLE) and FILE for input and output.
WRITE:  ?OPEN   CONSOLE  ; Open for Input/Output.
        WBR     ERROR    ; Error, process it.
        ?OPEN   FILE     ; Open (create?) FILE.
        WBR     ERROR    ; Error, process it.

        ; Write greeting and put I/O buffer byte pointer in the packet.
        ?WRITE  CONSOLE  ; Write message.
        WBR     ERROR    ; Error, process it.
        XLEFB   0,BUF*2  ; Get byte pointer to I/O buffer.
        XWSTA   0, CONSOLE+?IBAD ; Put in CON packet.

        ; Read line, check for terminator, write to FILE.
        NLDAI   'RD', 0  ; Put 'RD' terminator in AC0.
LOOP:   ?READ   CONSOLE  ; Read a line.
        WBR     ERROR    ; Error, process it.
        XNLDA   1, BUF   ; Get first word of buffer.
        WSNE    0,1      ; Did user type RD?
        WBR     SPOS     ;    Yes, do ?SPOS.
        ?WRITE  FILE     ;    No, write line to FILE.
        WBR     ERROR    ; Error, process it.
        WBR     LOOP     ; Get next line from user.

        ; Set position at beginning of FILE.
SPOS:   NLDAI   0, 1     ; Get 0 in AC1.
        XWSTA   1, FILE+?IRNH ; Put in record number word.
        XNLDA   2, FILE+?ISTI ; Get FILE's specifications.
        WMOV    2, 0          ; Save old specifications in AC0.
        WIORI   ?IPST,  2     ; Add ?IPST specification.
        XNSTA   2, FILE+?ISTI ; Put in FILE specifications.
        ?SPOS   FILE          ; Position at FILE start.
        WBR     ERROR         ; Error, process it.
        XNSTA   0, FILE+?ISTI ; Restore old specifications.
```

Figure B-9. Listing of Program WRITE.SR (Continues)

```
        ; Read lines back from FILE, write to terminal.
LOOP1:  ?READ   FILE      ; Read from FILE into buffer.
        WBR     EOF       ; Error, test for EOF.
        ?WRITE  CONSOLE   ; Write line to CON.
        WBR     ERROR     ; Error, process it.
        WBR     LOOP1     ; Read/Write another line.
EOF:    NLDAI   EREOF, 2  ; Code for End of File.
        WSEQ    0, 2      ; Skip if error is EOF.
        WBR     ERROR     ; Other error, process.
        ; CLOSE the file.
CLOSE:  ?CLOSE  CONSOLE   ; Close terminal.
        WBR     ERROR     ; Error, process it.
        ?CLOSE  FILE      ; Close FILE.
        WBR     ERROR     ; Error, process it.
        WSUB    2,2       ; Set good return flags.
        WBR     BYE       ; Skip error flags.
        ;  Process error and/or return here.
ERROR:  WLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
BYE:    ?RETURN           ; To CLI.
        WBR     ERROR     ; Return error.
        ; Open and I/O packet for terminal (minimum packet).
        .ENABLE WORD      ; Most entries = 16 bits.
        ; Open and I/O packet -- needed for I/O.
CONSOLE: .BLK   ?IBLT     ; Packet length.
        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO ; Data-sensitive records, Input/Output.
        .LOC    CONSOLE+?IMRS
         -1               ; Memory block size, default to 2048.
        .LOC    CONSOLE+?IBAD
        .DWORD  ITEXT*2   ; 2-word byte pointer to message and
                          ;    later to I/O buffer.
        .LOC    CONSOLE+?IRCL
        120.              ; Maximum record length of 120 characters.
        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2     ; 2-word byte pointer to filename.
        .LOC    CONSOLE+?IDEL
        .DWORD  -1        ; Data-sensitive delimiter table address;
                          ;    default to NL, CR, NUL.
        .LOC    CONSOLE+?IBLT ; Default others to 0.
        ; End of console I/O packet.
        ; Filename, buffer, messages.  .NOLOC 1 follows.
        .NOLOC  1         ; To save listing space.
CON:    .TXT "@CONSOLE"   ; Use generic name.
BUF:    .BLK 60.          ; 60. words, 120. characters.
ITEXT:  .TXT "I write lines to file FILE. Type
 RD[NL] to read lines back and stop.<12>"
        .NOLOC  0         ; Resume listing.
```

Figure B-9. Listing of Program WRITE.SR (Continued)

```
        ; Open and I/O packet for FILE. You can omit
        ;    entries set to 0 if you specify a packet length
        ;    of ?IBLT, as shown with .BLK and .LOC here.
FILE:   .BLK    ?IBLT     ; Packet length.
        .LOC    FILE+?ICH
        0                 ; For system-created channel number.
        .LOC    FILE+?ISTI
        ?OFCR+?OFCE+?ICRF+?RTDS+?OFIO ; Delete and recreate,
                          ; change format if needed, data-sensitive
                          ; records, input and output.
        .LOC    FILE+?ISTO
        0                 ; Default file type.
        .LOC    FILE+?IMRS
         -1               ; Memory block size, default to 2048.
        .LOC    FILE+?IBAD
        .DWORD  BUF*2     ; Buffer. Use same as CONSOLE.
        .LOC    FILE+?IRES
        0                 ; Tape density, default to VSGEN.
        .LOC    FILE+?IRCL
        120.              ; Maximum record length of 120 characters.
        .LOC    FILE+?IRLR
        0                 ; System returns number of chars. transferred.
        .LOC    FILE+?IRNW
        0                 ; Reserved for system.
        .LOC    FILE+?IRNH
        .DWORD  0         ; 2 words for record number, default to
                          ;    the next record.
        .LOC    FILE+?IFNP
        .DWORD  FNAME*2   ; 2-word byte pointer to filename.
        .LOC    FILE+?IDEL
        .DWORD  -1        ; Data-sensitive delimiter table address;
                          ;    default to NL, CR, NUL.
              ; Default other specifications.
        .LOC    FILE+?IBLT ; Packet length.
         ; End of FILE packet.
FNAME:  .TXT  "FILE"      ; Disk filename.

        .END    WRITE
```

Figure B–9. Listing of Program WRITE.SR (Concluded)

The assembly, link, and execute commands are

```
XEQ MASM WRITE
XEQ LINK WRITE
XEQ WRITE
```

Typical lines of dialog during the execution of program WRITE are next.

```
) XEQ WRITE
I write lines to file FILE. Type RD[NL] to read lines back and stop.
LINE 1 OF 3.
LINE 2 OF 3.
LINE 3 (AND LAST) OF 3.
RD
LINE 1 OF 3.
LINE 2 OF 3.
LINE 3 (AND LAST) OF 3.


) TYPE FILE
LINE 1 OF 3.
LINE 2 OF 3.
LINE 3 (AND LAST) OF 3.

)
```


Licensed Material - Property of Data General Corporation

Program Set 6 — DLIST.SR

Program DLIST gets a directory name from the CLI command line (?GTMES system call) and opens the directory and the line printer queue. Then, DLIST gets all filenames in the directory file and writes them to the line printer queue. The program includes I/O calls ?GNFN, ?OPEN, ?WRITE, and ?SEND.

Figure B-10 contains DLIST.SR.

```
; This program lists all filenames in a directory to the
;    line printer. It uses the ?GTMES system call to get the
;    directory name from the CLI command line and the system
;    call ?GOPEN to open the directory.  It shows system
;    calls ?GOPEN, ?OPEN, ?READ/?WRITE, ?GNFN, and ?SEND.
;    To execute it, type    X  prog-name  dir-name.
        .TITLE  DLIST
        .ENT    DLIST
        .NREL

        ; Get directory name, open directory file and the
        ;    line printer queue.
DLIST:  ?GTMES  CLIMSG   ; Get directory name argument.
        WBR     ERROR    ; Error, process it.
        LLEFB   0, DIRNAME*2 ; Get byte pointer to the directory name.
        NLDAI   -1, 1    ; -1 specifies system-assigned channel number
                         ;    for ?GOPEN.
        ?GOPEN  DIR      ; Open the directory.
        WBR     ERROR    ; Error, process it.
        ?OPEN   LINEP    ; Open @LPT queue.
        WBR     ERROR    ; Error, process it.
; Get next name via ?GNFN, write it to the printer.
        XNLDA   1, DIR+?ICH ; Keep channel number in AC1.
NEXT:   ?GNFN   GNAME    ; Put filename in ?GNFN buffer.
        WBR     EOF      ; Error, check for EOF.
        ?WRITE  LINEP    ; Write to LPT queue.
        WBR     ERROR    ; Error, process it.
        XLEFB   2, NL*2  ; Get address of NL character.
        XWSTA   2, LINEP+?IBAD ; Put it in LPT buffer.
        ?WRITE  LINEP    ; Write NL character to LPT queue.
        WBR     ERROR    ; Error, process it.
        XLEFB   2, FNAME*2  ; Get byte pointer to name buffer.
        XWSTA   2, LINEP+?IBAD ; Restore buffer address.
        WBR     NEXT     ; Get another filename.
EOF:    NLDAI   EREOF, 2 ; Code for End of File.
        WSEQ    0, 2     ; Skip if error was EOF.
        WBR     ERROR    ; Other error, process it.
```

Figure B-10. Listing of Program DLIST.SR (Continues)

```
        ; Done with filenames. Get ?SEND parameters, issue ?SEND.
        XLEFB   0,CNAME*2 ; Byte pointer to console name.
        XLEFB   1,TMSG*2 ; Byte pointer to ?SEND message.
        WLDAI   (CLIMSG-TMSG)*2+1S22,  2  ; Message length +
                        ; byte pointer flag (1S22).
        ?SEND           ; Send message to @CONSOLE.
        WBR     ERROR   ; Error, process it.
        WSUB    2, 2    ; Done. Set good return flags.
        WBR     BYE     ; Goodbye.

ERROR:  NLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
BYE:    ?RETURN                 ; To CLI.
        WBR     ERROR           ; Return error.

NL:     .TXT  "<12>"    ; Puts each name on new line.

        ; ?SEND console name and message.  .NOLOC 1 follows.

        .NOLOC  1
CNAME:  .TXT  "@CONSOLE" ; Use generic name.

TMSG:   .TXT  "All filenames written to LPT. Bye."

        .NOLOC  0       ; Resume listing all.

        .ENABLE WORD    ; Most packet entries = 16 bits.

        ; ?GTMES packet to get directory name from CLI command.

CLIMSG: .BLK    ?GTLN   ; ?GTMES packet length.
        .LOC    CLIMSG+?GREQ
        ?GARG   ; Put argument only in ?GRES.
        .LOC    CLIMSG+?GNUM
        1       ; Argument # 1 is directory name (0 is program name).
        .LOC    CLIMSG+?GRES
        .DWORD  DIRNAME*2 ; 2-word byte pointer to directory name buffer.
        .LOC    CLIMSG+?GTLN ; Default others to 0.
        ; End of ?GTMES packet.

DIRNAME: .BLK   50.     ; Buffer for directory pathname.

        ; ?OPEN packet, as needed for directory.

DIR:    .BLK    ?OPLT   ; ?GOPEN packet length.
                        ; No user specifications are needed
                        ;     for directory ?GOPEN.
        .LOC    DIR+?OPLT ; End of ?GOPEN packet.


        ; ?GNFN Packet to get next filename.

GNAME:  .BLK    ?NFLN   ; Packet length.
        .LOC    GNAME+?NFKY
        .DWORD  0       ; System uses the first 2 words.
        .LOC    GNAME+?NFNM
        .DWORD  FNAME*2  ; 2-word byte pointer to filename buffer.
        .LOC    GNAME+?NFTP
        .DWORD  -1      ; Template byte pointer, default, none.
        .LOC    GNAME+?NFLN ; End of ?GNFN pkt.

FNAME:  .BLK    16.     ; Maximum filename = 32 chars.
```

Figure B-10. Listing of Program DLIST.SR (Continued)

Licensed Material - Property of Data General Corporation

```
        ; Open, I/O Packet for printer (output) file.
LINEP:  .BLK    ?IBLT     ; Packet length.
        .LOC    LINEP+?ICH
        0                 ; System puts channel number here.
        .LOC    LINEP+?ISTI
        ?ICRF+?RTDS+?OFOT ; Data-sensitive records, output.
        .LOC    LINEP+?ISTO
        0                 ; Default file type.
        .LOC    LINEP+?IMRS
         -1               ; Memory block size, default to 2048.
        .LOC    LINEP+?IBAD
        .DWORD  FNAME*2   ; 2-word byte pointer to text/buffer.
        .LOC    LINEP+?IRCL
        136.              ; Maximum record length of 136 characters.
        .LOC    LINEP+?IRLR
        0                 ; System returns number of chars transferred.
        .LOC    LINEP+?IRNH
        .DWORD  0         ; 2 words for record number; default to next.
        .LOC    LINEP+?IFNP
        .DWORD  LPTNM*2   ; 2-word byte pointer to filename.
        .LOC    LINEP+?IDEL
        .DWORD  -1        ; Data-sensitive delimiter table address;
                          ;    default to NL, CR, NUL.
        ; Other specifications can be 0.
        .LOC    LINEP+?IBLT ; Packet length.
         ; End of LINEP packet.
LPTNM:  .TXT   "@LPT"     ; Printer queue filename
                          ;    (could be @CONSOLE).
        .END    DLIST
```

Figure B-10. Listing of Program DLIST.SR (Concluded)

The assembly, link, and execute commands are

```
XEQ MASM DLIST
XEQ LINK DLIST
XEQ DLIST
```

Output goes both to the line printer and to @CONSOLE. Typical line printer output is next in response to the CLI command

```
) XEQ DLIST =


RUNTIME.PR
RINGLOAD.ST
INRING.SR
...
FILCREATE.OB
```

Typical @CONSOLE output is next in response to the CLI command

```
) XEQ DLIST =

From Pid    49 : All filenames written to LPT. Bye.
)
```

Program Set 7 — NEWTASK.SR

In multitasked program NEWTASK, the program (initial) task creates another task, and then kills itself so that the newly created task receives exclusive control. NEWTASK issues system calls ?TASK, ?MYTID, and ?IDKIL.

Figure B-11 contains NEWTASK.SR.

```
; This program, NEWTASK, has the initial task
;    create a new task with priority 1 and ID 2.
;    The initial task opens the terminal, creates
;    the new task, announces its death, gets its
;    priority/ID, and kills itself. The new task
;    gets control, writes a message, and ?RETURNs.
;    (The last task cannot kill itself with a task
;    call.) The program shows calls ?TASK, ?MYTID,
;    and ?IDKIL.
        .TITLE  NEWTASK
        .ENT    NEWTASK
        .TSK    2           ; 2 tasks.
        .NREL               ; Unshared.
        ; Open terminal, create new task, kill self.
NEWTASK: ?OPEN  CONSOLE     ; For input and output.
        WBR     ERROR       ; Error, process it.
        ?TASK   TPACKET     ; Create new task,
                            ;    ID of 2, priority of 1.
        WBR     ERROR       ; Error, process it.
        ?WRITE  CONSOLE     ; Write swan song ...
        WBR     ERROR       ; Error, process it.
        ?MYTID              ; Get ID (AC0), priority (AC1).
        WBR     ERROR       ; Error, process it.
        WMOV    0, 1        ; Move ID into AC1 ...
        ?IDKIL              ;     ... and die.
        WBR     ERROR       ; Error on ?IDKIL.
        ; New task is now the only task.
NTASK:  XLEFB   0, NMSG*2 ; Get byte pointer to message.
        XWSTA   0, CONSOLE+?IBAD ; Put in I/O packet.
        ?WRITE  CONSOLE     ; Write message.
        WBR     ERROR       ; Error, process it.
        WSUB    2, 2        ; Set AC2 for good return.
        WBR     BYE         ; Go and return.
        ; Error handler.
ERROR:  NLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
BYE:    ?RETURN             ; To CLI.
        WBR     ERROR       ; Return error.
```

Figure B-11. Listing of Program NEWTASK.SR (Continues)

Licensed Material - Property of Data General Corporation

```
        ; Open and I/O packet for terminal.
        .ENABLE WORD ; Most entries = 16 bits.
CONSOLE: .BLK   ?IBLT    ; Packet length.
        .LOC    CONSOLE+?ISTI
        ?ICRF+?RTDS+?OFIO ; Data-sensitive records, I/O.
        .LOC    CONSOLE+?IMRS
         -1             ; Memory block size, default to 2048.
        .LOC    CONSOLE+?IBAD
        .DWORD  ITEXT*2  ; 2-word byte pointer to text/buffer.
        .LOC    CONSOLE+?IRCL
        120.            ; Maximum record length of 120 characters.
        .LOC    CONSOLE+?IFNP
        .DWORD  CON*2    ; 2-word byte pointer to filename.
        .LOC    CONSOLE+?IDEL
        .DWORD  -1       ; Data-sensitive delimiter table address; none.
        .LOC    CONSOLE+?IBLT ; End of I/O packet.
        ; Filename and messages.  .NOLOC 1 follows.
        .NOLOC  1
CON:    .TXT  "@CONSOLE"
ITEXT:  .TXT  "I'm the default task. Have opened
 @CONSOLE and am about to ?IDKIL myself.<12>"
NMSG:   .TXT  "I'm the new task, about to ?RETURN.<12>"
        .NOLOC  0        ; Resume listing all.
        ; ?TASK packet for new task.
TPACKET:  .BLK  ?DSLTH   ; Short packet length.
        .LOC    TPACKET+?DLNK
        1               ; Any nonzero value for short packet.
        .LOC    TPACKET+?DLNL
        0               ; Low order bits of ?DLNK, reserved.
        .LOC    TPACKET+?DLNKB
        .DWORD  0        ; ?DLNKB, ?DLNKBL are reserved.
        .LOC    TPACKET+?DPRI
        1               ; Priority is 1 (initial task's
                        ;    priority is 0).
        .LOC    TPACKET+?DID
        2               ; ID is 2 (initial task's ID is 1).
        .LOC    TPACKET+?DPC
         .DWORD NTASK    ; Task's starting address.
        .LOC    TPACKET+?DAC2
        .DWORD  0        ; No message for the new task.
        .LOC    TPACKET+?DSTB
        .DWORD  STACK    ; Stack address is STACK.
        .LOC    TPACKET+?DSFLT
        -1              ; Stack fault handler; default.
        .LOC    TPACKET+?DSSZ
        .DWORD  60.      ; Stack size is 60. words.
        .LOC    TPACKET+?DFLGS
        0               ; Flag word, reserved.
        .LOC    TPACKET+?DRES
        0               ; Reserved.
        .LOC    TPACKET+?DNUM
        1               ; Number of tasks is 1.
        .LOC    TPACKET+?DSLTH ; End of ?TASK packet.
STACK:  .BLK    60.      ; 60. word stack for task.
        .END    NEWTASK
```

Figure B-11. Listing of Program NEWTASK.SR (Concluded)

The assembly, link, and execute commands are

```
XEQ MASM NEWTASK
XEQ LINK NEWTASK
XEQ NEWTASK
```

The @CONSOLE output from the execution of program NEWTASK is next.

```
) XEQ NEWTASK
I'm the default task. Have opened @CONSOLE and am about to ?IDKIL myself.
I'm the new task, about to ?RETURN.

)
```

Program Set 8 — BOOMER.SR

BOOMER is a fast multitasked (two tasks) file copy program. System calls ?XMTW and ?REC synchronize the program's read and write operations. BOOMER copies an existing input file to a freshly created output file. Either or both files can be tape or disk. You specify the filenames as arguments to the CLI command that executes program BOOMER.

Much of BOOMER is shared code to allow for general use. BOOMER illustrates system calls ?GTMES, ?TASK, ?XMTW, ?REC, ?KILL, ?OPEN, ?READ, and ?WRITE. BOOMER illustrates principles of input/output and of multitasking.

Figure B-12 contains BOOMER.SR.

```
; This program, BOOMER, has the initial task read from
;    an input file and an output task write to an output
;    file. There are two buffers. The input task reads
;    into BUFFER1 and transmits its address to the
;    output task. The output task then writes BUFFER1
;    while the input task fills BUFFER2. The output task
;    writes BUFFER2 and the cycle repeats. Both tasks
;    have priority 0; the output task has an ID of 2.
; For general use, all but packet and buffer code
;    is in shared area (.NREL 1). Each buffer is
;    8,192 bytes long, but you can change BUFLENGTH
;    to suit your needs.
; The program uses the CLI ?GTMES mechanism to
;    get the output and input filenames from the
;    command line. It assumes that the input file
;    exists.  It deletes/recreates the output file
;    with dynamic record format. Execute the program
;    with a CLI command of the form
;
;    X  progname  outputfile  inputfile
;
; Examples are:
;
;   X  BOOMER  @MTB0:3  :UDD:UTIL:PARU.32.SR
;
;   X  BOOMER  FOO  :UDD:UTIL:PARU.32.SR
        .TITLE  BOOMER
        .ENT    BOOMER
        .TSK    2       ; 2 tasks
        .NREL   1       ; Shared code.
; Initial task uses ?GTMES to get output filename
; (2nd argument), opens it; repeats ?GTMES to get input
; filename (1st argument), opens it; creates output task.
```

Figure B-12. Listing of Program BOOMER.SR (Continues)

```
BOOMER: ?GTMES GPACKET    ; Get input filename.
        WBR     ERROR     ; Error, process it.
        LLEFB   0, FNAME*2  ; Get byte address of
                          ; ?GTMES-returned filename.
        LWSTA   0, INPUT+?IFNP ; Put in INPUT I/O packet.
        ?OPEN   INPUT     ; Open input file.
        WBR     ERROR     ; Error, process it.
        NLDAI   1, 0      ; Get 1 in AC0.
        LNSTA   0, GPACKET+?GNUM ; Specify arg1.
        ?GTMES  GPACKET   ; Get output filename.
        WBR     ERROR     ; Error, process it.
        LLEFB   0, FNAME*2 ; Get byte address of filename.
        LWSTA   0, OUTPUT+?IFNP ; Put in OUTPUT I/O packet.
        ?OPEN   OUTPUT    ; Open output file.
        WBR     ERROR     ; Error, process it.

        ?TASK   TPACKET   ; Create OUTPUT task.
        WBR     ERROR     ; Error, process it.

; Loop reads into BUFFER1, transmits it to output task,
;    reads into BUFFER2, transmits it to output task.
;    Message for output task is the buffer address.

READER: ?READ   INPUT     ; Read a buffer from input file.
        WBR     EOF?      ; Error, check for EOF.
        LLEF    0, MAILBOX  ; Get address of message.
        LWLDA   1, INPUT+?IBAD ; Message is buffer address.
        ?XMTW             ; Awaken output task.
        WBR     ERROR     ; Error, process it.

; Swap buffer byte pointers for next read.

        LLEFB   0, BUFFER1*2 ; Get byte pointer to BUFFER1.
        LLEFB   2, BUFFER2*2 ; Get byte pointer to BUFFER2.
        WSNE    1,2       ; Skip next instruction if BUFFER1
                          ;    was used for last READ.
        WMOV    0,2       ; Make BUFFER1 current buffer.
        LWSTA   2,INPUT+?IBAD ; Put current buffer byte pointer
                          ;    into INPUT packet.
        WBR     READER    ; Read into current buffer.

; On EOF, get number of characters read from INPUT packet
;    and make this number the buffer length for last ?XMTW.

EOF?:   NLDAI   EREOF, 1  ; Code for EOF (end of file).
        WSEQ    0, 1      ; Skip if not EOF.
        WBR     ERROR     ; Other error, process it.
        LLEF    0,MAILBOX ; Get address of message.
        LWLDA   1,INPUT+?IBAD  ; Message is buffer byte pointer.
        WMOV    1,2       ; Copy to AC2 for indexing.
        NLDAI   -1,3      ; Put -1 in AC3.
        WLSH    3,2       ; Make buffer byte pointer a word pointer.
        LNLDA   3,INPUT+?IRLR ; Get number of characters read
                          ;    from INPUT I/O packet.
        LWSTA   3,-2,2    ; Make BUFLENGTH (AC2-2) the number
                          ;    of characters read.
        ?XMTW             ; Send the last buffer.
        WBR     ERROR     ; Error, process it.
        ?KILL             ; Input is done; output task will ?RETURN.

; Error handler.

ERROR:  WLDAI   ?RFEC+?RFCF+?RFER,  2  ; Error flags.
        ?RETURN           ; To CLI.
        WBR     ERROR     ; Return error.
```

Figure B-12. Listing of Program BOOMER.SR (Continued)

Licensed Material - Property of Data General Corporation

```
; Output task does the writing.
WRITER: LLEF    0,MAILBOX ; Get message address.
        ?REC              ; Wait for message.
        WBR     ERROR     ; Error, process it.
        LWSTA   1, OUTPUT+?IBAD  ; Got message.  It was a buffer byte
                                 ;    pointer; put it in I/O packet.
        WMOV    1,2       ; Copy to AC2 for indexing.
        NLDAI   -1,3      ; Put -1 in AC3.
        WLSH    3,2       ; Make byte pointer into word pointer.
        XWLDA   0,-2,2    ; Get buffer length left by input task
                          ;    (original length unless task hit EOF).
        LNSTA   0,OUTPUT+?IRCL ; Make this length the maximum record
                                 ;    length in the I/O packet.
        ?WRITE  OUTPUT    ; Write buffer to output file.
        WBR     ERROR     ; Error, process it.
        WLDAI   BUFLENGTH, 1 ; Get original buffer length.
        WSNE    0,1       ; Is current buffer length the same as the
                          ;    original buffer length?
        WBR     WRITER    ;    Yes, get another buffer.
        WSUB    2,2       ;    No. Done. Set for good return.
        ?RETURN           ; Return to CLI.
        WBR     ERROR     ; Return error.
; Buffers, message, packets in unshared code.
        .NREL
; Buffer declarations.
BUFLENGTH =     16384.    ; Need change only this.
        BUFLENGTH         ; For residual characters after EOF.
BUFFER1: .BLK   (BUFLENGTH+1)/2   ; Size of BUFFER1.
        BUFLENGTH         ; For residual characters after EOF.
BUFFER2: .BLK   (BUFLENGTH+1)/2   ; Size of BUFFER2.
; Mailbox for message.
MAILBOX: 0
        .ENABLE WORD      ; Most packet items = 16 bits.
; ?GTMES packet to get Input and Output filenames.
GPACKET: .BLK   ?GTLN     ; ?GTMES packet length.
        .LOC    GPACKET+?GREQ
        ?GARG             ; Put argument only in ?GRES.
        .LOC    GPACKET+?GNUM
        2                 ; Argument 2 is input filename.
        .LOC    GPACKET+?GRES
        .DWORD  FNAME*2   ; 2-word byte pointer to filename buffer.
        .LOC    GPACKET+?GTLN ; Default others to 0.
        ; End of ?GTMES packet.
```

Figure B–12. Listing of Program BOOMER.SR (Continued)

```
; Open and I/O packet for INPUT task.
INPUT:  .BLK    ?IBLT     ; Packet length.
        .LOC    INPUT+?ISTI
        ?ICRF+?RTDY+?OFIN ; Change format + Dynamic + Input.
        .LOC    INPUT+?IMRS
         -1               ; Memory block size, default to 2048.
        .LOC    INPUT+?IBAD
         .DWORD  BUFFER1*2 ;  Original input buffer.
        .LOC    INPUT+?IRCL
        BUFLENGTH         ; Record length is BUFLENGTH.
        .LOC    INPUT+?IRLR
        0                 ; System returns number of chars. transferred.
        .LOC    INPUT+?IFNP
        .DWORD  FNAME*2   ; 2-word byte pointer to filename.
        .LOC    INPUT+?IDEL
        .DWORD  -1        ; Data-sensitive delimiter table address;
                          ;    not relevant.
        .LOC    INPUT+?IBLT ; Packet length.
         ; End of INPUT packet.
; ?TASK packet for output task - minimum packet.
TPACKET: .BLK   ?DSLTH    ; Short packet length.
        .LOC    TPACKET+?DLNK
        1                 ; Any nonzero value for short packet.
        .LOC    TPACKET+?DPRI
        0         ; Priority is 0, as is input task's.
        .LOC    TPACKET+?DID
        2         ; ID is 2 (initial task's ID is 1).
        .LOC    TPACKET+?DPC
         .DWORD WRITER    ; Task's starting address.
        .LOC    TPACKET+?DSTB
        .DWORD  STACK     ; Stack address is STACK.
        .LOC    TPACKET+?DSFLT
        -1                ; Stack fault handler; default.
        .LOC    TPACKET+?DSSZ
        .DWORD  60.       ; Stack size is 60. words.
        .LOC    TPACKET+?DNUM
        1                 ; Number of tasks is 1.
        .LOC    TPACKET+?DSLTH ; End of ?TASK packet.
```

Figure B-12. Listing of Program BOOMER.SR (Continued)

```
; Open and I/O packet for OUTPUT task.
OUTPUT: .BLK    ?IBLT     ; Packet length.
        .LOC    OUTPUT+?ISTI
        ?OFCR+?OFCE+?ICRF+?RTDY+?OFIO ; Delete + create, change format,
                                      ;    dynamic records, Input/Output.
        .LOC    OUTPUT+?IMRS
         -1               ; Memory block size, default to 2048.
        .LOC    OUTPUT+?IBAD
        .DWORD  BUFFER1*2 ; Start output from BUFFER1.
        .LOC    OUTPUT+?IRCL
        BUFLENGTH         ; Record length of BUFLENGTH chars.
        .LOC    OUTPUT+?IRLR
        0                 ; System returns number of characters
                          ;    transferred here.
        .LOC    OUTPUT+?IFNP
        .DWORD  FNAME*2   ; 2-word byte pointer to filename.
        .LOC    OUTPUT+?IDEL
        .DWORD  -1        ; Data-sensitive delimiter table address;
                          ;    not relevant.
        .LOC    OUTPUT+?IBLT ; Default others to 0.
        ; End of Output I/O packet.
FNAME:  .BLK  (?MXPL+1)/2 ; Filename  buffer; system limit for
                          ;    number of characters.
STACK:  .BLK    60.       ; 60.-word task stack.
        .END    BOOMER
```

Figure B–12. Listing of Program BOOMER.SR (Concluded)

The assembly, link, and execute commands are

XEQ MASM BOOMER
XEQ LINK BOOMER
XEQ BOOMER outputfilename inputfilename

A typical CLI command to execute BOOMER.PR is next. The command places a copy of (input) file MINE.DATA into (output) file YOURS.DATA. File YOURS.DATA has dynamic records.

```
) XEQ BOOMER YOURS.DATA MINE.DATA

)
```

Program Set 9 — TIMEOUT.SR

TIMEOUT is a simple program that uses ?GTMES to get a number from the CLI command line that executes TIMEOUT, and then delays itself for this number of seconds. The number must be between 0 and 20., inclusive. Other examples of system call ?GTMES appear in programs DLIST.SR and BOOMER.SR in this appendix.

Figure B-13 contains TIMEOUT.SR.

```
; This program uses the CLI ?GTMES mechanism to get a number from
;    the CLI command that executed the program. Then it delays itself
;    for the given number of seconds. To run it, give the CLI command
;
;    X TIMEOUT secs
;
;    where    secs    is a decimal number from 0 through 20.
        .TITLE  TIMEOUT
        .ENT    TIMEOUT
        .NREL               ; Unshared.
; Use CLI ?GTMES to get number of secs. Puts
; ASCII value in AC2, binary value in AC1.
TIMEOUT: ?GTMES CLIMSG    ; Get number of seconds.
        WBR     ERROR     ; Error, process it.
; Check range of argument (returned in AC1).
        WCLM    1, 1      ; If not between values ...
        0                 ;    ... lower limit of 0 ...
        20.               ;    ... upper limit of 20..
        WBR     BADVALUE  ;    exit with "illegal" msg.
        NLDAI   1000., 0  ; Put 1000. in AC0.
        WMUL    1, 0      ; Get number of milliseconds in AC0.
        ?WDELAY           ; Delay for specified number of seconds.
        WBR     ERROR     ; Error, process it.
        WSUB    2, 2      ; Set for good return.
        WBR     BYE       ; Goodbye.
ERROR:  NLDAI ?RFEC+?RFCF+?RFER, 2    ; Error flags.
BYE:    ?RETURN                       ; To CLI.
        WBR     ERROR                 ; Return error.
BADVALUE: XLEFB 1, BMSG*2             ; Byte pointer to message.
        NLDAI (CLIMSG-BMSG)*2+?RFCF, 2 ; Length + flags.
        WBR     BYE                   ; Bye.
BMSG:   .TXT  "You gave an illegal delay."
```

Figure B-13. Listing of Program TIMEOUT.SR (Continues)

Licensed Material - Property of Data General Corporation

```
        ; ?GTMES packet to get number of seconds from the CLI command.
        .ENABLE WORD      ; Most entries = 16 bits.
CLIMSG: .BLK    ?GTLN     ; ?GTMES packet length.
        .LOC    CLIMSG+?GREQ
        ?GARG             ; Put argument only in ?GRES.
        .LOC    CLIMSG+?GNUM
        1                 ; Argument 1 is number of seconds
                          ;    (Argument 0 is programname).
        .LOC    CLIMSG+?GRES
        .DWORD  -1        ; No buffer needed.
        .LOC    CLIMSG+?GTLN ; Default others; end packet.
        .END    TIMEOUT
```

Figure B–13. Listing of Program TIMEOUT.SR (Concluded)

The assembly, link, and execute commands are

```
XEQ MASM TIMEOUT
XEQ LINK TIMEOUT
XEQ TIMEOUT
```

Typical lines of dialog during the execution of program TIMEOUT are next.

```
) XEQ TIMEOUT 3
```

(3 seconds elapse.)

```
) XEQ TIMEOUT 22
```

You gave an illegal delay.

```
)
```

Program Set 10 — DIRCREATE.F77 and CHECK.F77

DIRCREATE.F77 is a FORTRAN 77 program that, at runtime, is a process that creates directory file NEW_DIR and makes it the process's working directory. The process creates system calls ?CREATE (which requires a parameter packet) and ?DIR (which does not require a parameter packet). The process also creates system call ?RETURN (which does not require a parameter packet).

These system calls have the same effects as the respective CLI commands

```
CREATE/DIRECTORY NEW_DIR
DIR NEW_DIR
```

It's worth noting that the CLI program is a process named CLI.PR that accepts these two character strings, one at a time, and interprets them. Then, the CLI loads accumulators, creates any necessary parameter packets, and creates the respective system calls ?CREATE and ?DIR.

The point of this section is to list the general steps of writing FORTRAN 77 programs that create explicit system calls. (All FORTRAN 77 programs create implicit system calls; e.g., a TRUNCATE statement results in the runtime execution of system call ?TRUNCATE.) The section also contains a comprehensive example that you can easily generalize from. This section is a condensation of the chapter in the *FORTRAN 77 Environment Manual (AOS/VS)* that explains the system interface function, ISYS.

General Steps

The general steps are as follows:

1. List the system calls that your program creates and the PARU symbols that your program needs. The system call mnemonics, such as ?RETURN, are in file SYSID.32.SR. Place these symbols in a file whose name is based on the program name. An example for program MINE.F77 might be MINE_SYMBOLS, the contents of which are

   ```
   ?RETURN
   ?RFCF
   ?RFER
   ?RFEC
   ```

   Here, program MINE creates system call ?RETURN and uses PARU symbols ?RFCF, ?RFER, and ?RFEC. Make such a list for each source program unit that creates system calls.

Licensed Material - Property of Data General Corporation

2. Use program F77BUILD_SYM to read these symbols and create a file to include (via an %INCLUDE compiler directive or an INCLUDE statement) in your program. Do this for each source program unit that makes system calls. To continue the example in the previous step: F77BUILD_SYM reads symbol ?RFEC from file MINE_SYMBOLS and transforms it into the statements

```
INTEGER*2   ISYS_RFEC
PARAMETER (ISYS_RFEC = 4096)          ! ?RFEC = 10000K
```

Although F77BUILD_SYM places the statements in file QSYM.F77.IN, you rename the file to MINE.F77.IN.

F77BUILD_SYM also requires as input files SYSID.32.LS and PARU.32.LS. You use the Macroassembler to create these files.

It's possible to build file QSYM.F77.IN with all the system call names in SYSID.32 and all the symbols in PARU.32. You could then have all your F77 programs that make explicit system calls include QSYM.F77.IN. This method eliminates the need for program-specific files such as MINE_SYMBOLS and MINE.F77.IN. However, it has its drawbacks. It results in several thousand extra statements (INTEGER and PARAMETER) for the compiler to process, larger object files, and larger program files.

3. Write your programs, being sure to have appropriate %INCLUDE compiler directives and INCLUDE statements. Compile, link, and test the resulting software.

Program Listings

Six related files appear next. Their figure numbers, names, and descriptions are as follows.

Figure Number	Filename	File Description
B-14	DIRCREATE.F77	Program to make system calls ?CREATE and ?DIR
B-15	DIRCREATE_SYMBOLS	Symbols in files SYSID.32 and PARU.32
B-16	DIRCREATE_SYMBOLS.F77.IN	Symbols and values from SYSID.32 and PARU.32; included in DIRCREATE.F77
B-17	CHECK.F77	Subroutine to report errors that occurred during a system call
B-18	CHECK_SYMBOLS	Symbols in files SYSID.32 and PARU.32
B-19	CHECK_SYMBOLS.F77.IN	Symbols and values from SYSID.32 and PARU.32; included in CHECK.F77

Figure B-14 contains program DIRCREATE.F77.

```
      PROGRAM DIRCREATE
C     This program is a complete example program that makes
C        system calls ?CREATE (with a parameter packet) and
C        ?DIR (without a parameter packet).
      INTEGER*4 AC0, AC1, AC2 ! Accumulators
      CHARACTER*8 NEW_DIR / 'NEW_DIR<0>' /
      INTEGER*4 RESULT_CODE   ! Result of system call
C     Define the parameter packet for ?CREATE.
      INTEGER*2 CREATE_PACKET(0:9)      ! Ten 16-bit words
      INTEGER*2 ?CFTYP          ! offset ?CFTYP, single word
      EQUIVALENCE (CREATE_PACKET(0),   ?CFTYP)
      INTEGER*2 ?CHFS           ! offset ?CHFS,  single word
      EQUIVALENCE (CREATE_PACKET(1),   ?CHFS)
      INTEGER*4 ?CTIM           ! offset ?CTIM,  double word
      EQUIVALENCE (CREATE_PACKET(2),   ?CTIM)
      INTEGER*4 ?CACP           ! offset ?CACP,  double word
      EQUIVALENCE (CREATE_PACKET(4),   ?CACP)
      INTEGER*4 ?CMSH           ! offset ?CMSH,  double word
      EQUIVALENCE (CREATE_PACKET(6),   ?CMSH)
      INTEGER*2 ?CMIL           ! offset ?CMIL,  single word
      EQUIVALENCE (CREATE_PACKET(10K), ?CMIL)
C or  EQUIVALENCE (CREATE_PACKET( 8 ), ?CMIL)
      INTEGER*2 ?CMRS           ! offset ?CMRS,  single word
      EQUIVALENCE (CREATE_PACKET(11K), ?CMRS)
C or  EQUIVALENCE (CREATE_PACKET( 9 ), ?CMRS)
```

Figure B-14. Listing of Program DIRCREATE.F77 (Continues)

```
C     File DIRCREATE_SYMBOLS.F77.IN contains, from files SYSID.32
C        and PARU.32, symbols and values for this program.
      INCLUDE 'DIRCREATE_SYMBOLS.F77.IN'
C     Load AC0 with the byte address of the new directory's name.
      AC0 = BYTEADDR(NEW_DIR)
      AC1 = 0
C     Prepare the parameter packet for ?CREATE.
      ?CFTYP = 0*400K + ISYS_FDIR  ! Left byte is 0, right byte is ?FDIR
      ?CHFS = -1
      ?CTIM = -1
      ?CACP = -1
      ?CMSH = 0
      ?CMIL = -1
      ?CMRS = 0
      AC2 = WORDADDR(CREATE_PACKET)
C     Make system call ?CREATE, according to accumulator values
C        and its parameter packet, via function subprogram ISYS.
      RESULT_CODE = ISYS(ISYS_CREATE, AC0, AC1, AC2)
C     Report any errors that occurred during ?CREATE.
      CALL CHECK (RESULT_CODE, 'While creating directory NEW_DIR')
C     Set up accumulators for system call ?DIR.
      AC0 = BYTEADDR(NEW_DIR)
      AC1 = 0
      AC2 = 0
C     Make system call ?DIR, according to accumulator values,
C        via function subprogram ISYS.
      RESULT_CODE = ISYS(ISYS_DIR, AC0, AC1, AC2)
C     Report any errors that occurred during ?DIR.
      CALL CHECK (RESULT_CODE, 'While moving to directory NEW_DIR')
      STOP
      END
```

Figure B-14. Listing of Program DIRCREATE.F77 (Concluded)

Figure B-15 contains file DIRCREATE_SYMBOLS.

```
?CREATE

?DIR

?FDIR
```

Figure B-15. File DIRCREATE_SYMBOLS

Figure B–16 contains file DIRCREATE_SYMBOLS.F77.IN. Program F77BUILD_SYM creates QSYM.F77.IN from file DIRCREATE_SYMBOLS; you rename QSYM.F77.IN to DIRCREATE_SYMBOLS.F77.IN.

```
****  F77 INCLUDE file for system parameters ****

****  INTEGER*4 parameters for SYSID  ****
      INTEGER*4  ISYS_CREATE
      PARAMETER (ISYS_CREATE = 0)    ! ?.CREATE = OK
      INTEGER*4  ISYS_DIR
      PARAMETER (ISYS_DIR = 61)      ! ?.DIR = 75K
****  Parameters for PARU   ****
      INTEGER*2  ISYS_FDIR
      PARAMETER (ISYS_FDIR = 10)     ! ?FDIR = 12K
****  END of F77 INCLUDE file for system parameters ****
```

Figure B–16. File DIRCREATE_SYMBOLS.F77.IN

Suppose you give the CLI commands

```
DELETE/2=IGNORE DIRCREATE.LS
F77/L=DIRCREATE.LS DIRCREATE.F77
QPRINT DIRCREATE.LS
```

Then, the contents of DIRCREATE_SYMBOLS.F77.IN appear on paper just after the statement

```
INCLUDE 'DIRCREATE_SYMBOLS.F77.IN'
```

Figure B–17 contains subroutine subprogram CHECK.F77. This is a general–purpose routine that you can call from any program that invokes the system interface function ISYS. Suppose CHECK receives an error code of 0 in its parameter ECODE. CHECK simply returns, because an error code of 0 means that the system call executed without an error. If CHECK receives a nonzero error code, it responds by executing ?RETURN and displaying whatever message CHECK has received in its parameter TEXT. You might want to change CHECK's response to a nonzero error code to meet your installation's needs.

```
      SUBROUTINE CHECK (ECODE, TEXT)
      INTEGER*4 ECODE           ! Error code from main program,
                                !   returned there from ISYS
      CHARACTER*(*) TEXT        ! Error text from main program
                                !   to accompany ECODE
      INTEGER*4 AC2             ! Accumulator 2
      INTEGER*4 RESULT          ! of ?RETURN call
      INCLUDE 'CHECK_SYMBOLS.F77.IN'
      IF (ECODE .EQ. 0) THEN
              RETURN            ! ISYS executed without an error.
      END IF
C     ISYS executed with an error, so report it.
      AC2 =    ISYS_RFCF        ! CLI format
     *       + ISYS_RFER        ! Error -- Severity = 2
     *       + ISYS_RFEC        ! AC0 contains the error code
      AC2 = AC2 + MIN(LEN(TEXT),255)
C     Execute ?RETURN and report the error from ISYS.
      RESULT = ISYS(ISYS_RETURN, ECODE, BYTEADDR(TEXT), AC2)
      STOP '- Impossible-to-occur error occurred during ?RETURN'
      END
```

Figure B-17. Subroutine CHECK.F77

Figure B-18 contains file CHECK_SYMBOLS.

```
?RETURN
?RFCF
?RFER
?RFEC
```

Figure B-18. File CHECK_SYMBOLS

Figure B-19 contains file CHECK_SYMBOLS.F77.IN. Program F77BUILD_SYM creates QSYM.F77.IN from file CHECK_SYMBOLS; you rename QSYM.F77.IN to CHECK_SYMBOLS.F77.IN.

```
****  F77 INCLUDE file for system parameters ****
****  INTEGER*4 parameters for SYSID  ****
      INTEGER*4  ISYS_RETURN
      PARAMETER (ISYS_RETURN = 200)                    ! ?.RETURN = 310K
****  Parameters for PARU   ****
      INTEGER*2  ISYS_RFCF
      PARAMETER (ISYS_RFCF = -32768)   ! ?RFCF = 100000K

      INTEGER*2  ISYS_RFEC
      PARAMETER (ISYS_RFEC = 4096)     ! ?RFEC = 10000K

      INTEGER*2  ISYS_RFER
      PARAMETER (ISYS_RFER = 16384)    ! ?RFER = 40000K

****  END of F77 INCLUDE file for system parameters ****
```

Figure B-19. File CHECK_SYMBOLS.F77.IN

Suppose you give the CLI commands

```
DELETE/2=IGNORE CHECK.LS
F77/L=CHECK.LS CHECK.F77
QPRINT CHECK.LS
```

Then, the contents of CHECK_SYMBOLS.F77.IN appear on paper just after the statement

```
INCLUDE 'CHECK_SYMBOLS.F77.IN'
```

Program Construction

First, be sure you have at least Read access to files SYSID.32.LS and PARU.32.LS. Program F77BUILD_SYM, to which you must have at least Execute access, requires these files. Typically, your system manager creates them in :UTIL via executing the Macroassembler. The *FORTRAN 77 Environment Manual (AOS/VS)* explains how to create these listing files.

Next is dialog that shows the construction of program file DIRCREATE.PR from files DIRCREATE.F77 (Figure B-14), DIRCREATE_SYMBOLS (Figure B-15), DIRCREATE_SYMBOLS.F77.IN (Figure B-16), CHECK.F77 (Figure B-17), CHECK_SYMBOLS (Figure B-17), and CHECK_SYMBOLS.F77.IN. All filename suffixes appear for emphasis; you frequently can ignore them during commands to the compiler and Link. The output from the compiler and Link programs doesn't appear in the dialog.

```
) COMMENT Create QSYM.F77.IN from DIRCREATE_SYMBOLS.
) XEQ F77BUILD_SYM DIRCREATE_SYMBOLS
) DELETE/2=IGNORE DIRCREATE_SYMBOLS.F77.IN
) RENAME QSYM.F77.IN DIRCREATE_SYMBOLS.F77.IN
) COMMENT DIRCREATE_SYMBOLS.F77.IN is ready for inclusion
) COMMENT    by DIRCREATE.F77.
) COMMENT Create DIRCREATE.OB from DIRCREATE.F77 and
) COMMENT    DIRCREATE_SYMBOLS.F77.IN.
) F77 DIRCREATE.F77
) COMMENT
) COMMENT Create QSYM.F77.IN from CHECK_SYMBOLS.
) XEQ F77BUILD_SYM CHECK_SYMBOLS
) DELETE/2=IGNORE CHECK_SYMBOLS.F77.IN
) RENAME QSYM.F77.IN CHECK_SYMBOLS.F77.IN
) COMMENT CHECK_SYMBOLS.F77.IN is ready for inclusion by CHECK.F77.
) COMMENT Create CHECK.OB from CHECK.F77 and CHECK_SYMBOLS.F77.IN.
) F77 CHECK.F77
) COMMENT CREATE DIRCREATE.PR from DIRCREATE.OB and CHECK.OB
) F77LINK DIRCREATE.OB CHECK.OB
) COMMENT Program file DIRCREATE.PR is now ready for execution.
```

Program Execution

Execute FORTRAN 77 programs that make explicit system calls in the same way that you execute any other FORTRAN 77 program.

Next is the dialog that shows the execution of program file DIRCREATE.PR. Read it carefully to see what happens when NEW_DIR does not exist prior to DIRCREATE.PR's execution, and then what happens when NEW_DIR does exist prior to DIRCREATE.PR's execution.

```
) COMMENT EXECUTE DIRCREATE.PR after making sure that NEW_DIR
) COMMENT    doesn't exist.
) DELETE/2=IGNORE NEW_DIR
) DIRECTORY

:UDD4:ALICE

) DATE; TIME

26-AUG-85
16:38:37

) XEQ DIRCREATE.PR

STOP

) COMMENT There were no errors when ?CREATE and ?DIR executed,
) COMMENT    so subroutine CHECK reported no error.
) FILESTATUS/ASSORT NEW_DIR

DIRECTORY :UDD4:ALICE

  NEW_DIR          DIR   26-AUG-85   16:38:44        0

) DIRECTORY

:UDD4:ALICE

) COMMENT The ?DIR call in DIRCREATE.PR executed successfully, but
) COMMENT    NEW_DIR was the current directory only during the rest
) COMMENT    of the life of process DIRCREATE.PR. When the process
) COMMENT    terminated, the directory, :UDD4:ALICE, of the parent
) COMMENT    process once again became the current directory.
) DIRECTORY NEW_DIR
) FILESTATUS

) COMMENT As expected, the FILESTATUS command returns no information
) COMMENT    because directory NEW_DIR is empty.
) DIRECTORY ^
) COMMENT We're back in directory ALICE. Execute DIRCREATE.PR again,
) COMMENT    and see what happens when ?CREATE tries to create a file
) COMMENT    (NEW_DIR) that already exists.
) XEQ DIRCREATE.PR

*ERROR*
FILE NAME ALREADY EXISTS
While creating directory NEW_DIR
ERROR: FROM PROGRAM
XEQ,DIRCREATE.PR

) COMMENT Notice how subroutine CHECK reported the error and
) COMMENT    terminated the process.
```

Program Set 11 — CREATE_WINDOW.SR.

Program CREATE_WINDOW uses system call ?WINDOW to create a character window. After creating and setting up the window, the program uses system call ?PROC to pass the window to a CLI son process. After the CLI process terminates, CREATE_WINDOW deletes the window.

Figure B-20 contains CREATE_WINDOW.SR.

```
        .TITLE CREATE_WINDOW
        .ENT CREATE_WINDOW
        .NREL 0

; This program creates a character window, gets its status, adjusts its
;   appearance, turns its permanence on, and makes the window visible to
;   the user.  The program then ?PROCs a CLI into the window and pends until
;   the user terminates the CLI.  The program then turns off the window's
;   permanence and deletes the window.

; The program assumes that a window already exists on the pixel map (AOS/VS
;   creates a default window during system initialization), and that the
;   @CONSOLE file is the pixel map.

CREATE_WINDOW:
        ; Create a character window using ?WINDOW system call,
        ; function code ?WIN_CREATE_WINDOW.

CREATE:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet into AC2.
        NLDAI   ?WIN_CREATE_WINDOW,0            ; Specify ?WIN_CREATE_WINDOW
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        WLDAI   1S?WIN_PKT.FLAGS.IN_PATH,0      ; Specify window pathname
        XWSTA   0,?WIN_PKT.FLAGS,2              ; by setting flag in main packet.
        XLEFB   0,PMAP*2                        ; Store byte pointer to pathname
        XWSTA   0,?WIN_PKT.PATH,2               ; in main packet.
        NLDAI   PMAP_PATH_LENGTH,0              ; Store length of pathname in
        XNSTA   0,?WIN_PKT.PATH_BUF_LEN,2       ; main packet.
        XLEF    0,WIN_CRE                       ; Store word pointer to subpacket
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; in main packet.
        WSUB    0,0                             ; AC0 must be set to zero.
        WSUB    1,1                             ; AC1 must be set to zero.

        ?WINDOW                                 ; Create a window.
        WBR     ERROR                           ; Quit.

        ; Use window ID as window specifier for remaining ?WINDOW calls.

        XWLDA   0,?WIN_PKT.WIND_ID,2            ; Load window ID and
        XWSTA   0,WIND_ID                       ; store it in location WIND_ID

        WLDAI   1S?WIN_PKT.FLAGS.IN_WIND_ID,0   ; Use window ID as specifier,
        XWSTA   0,?WIN_PKT.FLAGS,2              ; by setting flag in main packet.
        WSUB    0,0                             ; Clear AC0.
        XWSTA   0,?WIN_PKT.PATH,2               ; Clear pathname in main packet.
        XNSTA   0,?WIN_PKT.PATH_BUF_LEN,2       ; Clear pathname length.

        ; Get window status information using function code
        ; ?WIN_WINDOW_STATUS.

STATUS:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet in AC2.
        NLDAI   ?WIN_WINDOW_STATUS,0            ; Specify ?WIN_WINDOW_STATUS
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        XLEF    0,WIN_STAT                      ; Store word pointer to subpacket
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; in main packet.
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.

        ?WINDOW                                 ; Get status. System returns
                                                ; information in subpacket.
        WBR     ERROR                           ; Quit.
```

Figure B-20. Listing of Program CREATE_WINDOW.SR (Continues)

```
        ; Adjust the appearance of the window using the ?WINDOW system call,
        ; function code ?WIN_DEFINE_PORTS.
DEF_PORTS:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet in AC0.
        NLDAI   ?WIN_DEFINE_PORTS,0             ; Specify ?WIN_DEFINE_PORTS
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        XLEF    0,WIN_DPORT                     ; Store word pointer to subpacket
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; in main packet.
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.
        ?WINDOW                                 ; Define view port and scan port.
        WBR     ERROR                           ; Quit.

        ; Turn permanence on for the window using the ?WINDOW system call,
        ; function code ?WIN_PERMANENCE_ON.
PERM_ON:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet in AC0.
        NLDAI   ?WIN_PERMANENCE_ON,0            ; Specify ?WIN_PERMANENCE_ON
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; No subpacket for this function.
        ?WINDOW                                 ; Turn permanence on.
        WBR     ERROR                           ; Quit.

        ; Make the window visible to the user using the ?WINDOW system call,
        ; function code ?WIN_UNHIDE_WINDOW.
UNHIDE:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet in AC0.
        NLDAI   ?WIN_UNHIDE_WINDOW,0            ; Specify ?WIN_UNHIDE_WINDOW
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; No subpacket for this function.
        ?WINDOW                                 ; Unhide the window.
        WBR     ERROR                           ; Quit.

        ; ?PROC a CLI, passing the window as the CLI's @CONSOLE file.
        ; AOS/VS assigns the window to the CLI.
PROC_CLI:
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.
        XLEF    2,PROC_PKT                      ; Load word pointer to ?PROC
                                                ; packet in AC0.
        ?PROC                                   ; ?PROC a CLI in window.
        WBR     ERROR                           ; Quit.

        ; The program pends here until the user terminates the CLI.
        ; Unless there is an error, the program then turns permanence off
        ; and deletes the window.

        ; Turn permanence off for the window using the ?WINDOW call,
        ; function code ?WIN_PERMANENCE_OFF.
PERM_OFF:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet in AC0.
        NLDAI   ?WIN_PERMANENCE_OFF,0           ; Specify ?WIN_PERMANENCE_OFF
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; No subpacket for this function.
        ?WINDOW                                 ; Turn permanence off.
        WBR     ERROR                           ; Quit.
```

Figure B-20. Listing of Program CREATE_WINDOW.SR (Continued)

Licensed Material - Property of Data General Corporation

```
        ; Delete the window using the ?WINDOW system call,
        ; function code ?WIN_DELETE_WINDOW.
DELETE:
        XLEF    2,WIN_PKT                       ; Load word pointer to main
                                                ; packet in AC0.
        NLDAI   ?WIN_DELETE_WINDOW,0            ; Specify ?WIN_DELETE_WINDOW
        XNSTA   0,?WIN_PKT.FUNC,2               ; function code in main packet.
        WSUB    0,0                             ; Clear AC0.
        WSUB    1,1                             ; Clear AC1.
        XWSTA   0,?WIN_PKT.SUBPKT,2             ; No subpacket for this function.
        ?WINDOW                                 ; Delete the window.
        WBR     ERROR                           ; Quit.
        WSUB    2,2                             ; Set flags for normal return.
        WBR     BYE                             ; Branch to good return.
        ; Return to the calling process.
ERROR:  WLDAI   ?RFEC!?RFCF!?RFER,2             ; Error flags:
                                                ; Error code is in AC0 (?RFEC),
                                                ; message is in CLI format
                                                ; (?RFCF), caller should handle
                                                ; this as an error (?RFER).
BYE:    ?RETURN                                 ; Return to CLI.
        WBR     ERROR                           ; ?RETURN error return.
        ; Set up a place to save the window ID.
WIND_ID: 0
        ; Set up the main ?WINDOW packet
WIN_PKT: .BLK   ?WIN_PKT_LEN                    ; Allocate enough space for
                                                ; ?WINDOW packet.
        .LOC    WIN_PKT+?WIN_PKT.PKT_ID         ; Packet ID.
        .DWORD  ?WIN_PKT_PKTID
        .LOC    WIN_PKT+?WIN_PKT.FUNC           ; Function code.
        .WORD   0
        .LOC    WIN_PKT+?WIN_PKT.CHAN_NUM       ; Channel number of
                                                ; target window.
        .WORD   0
        .LOC    WIN_PKT+?WIN_PKT.FLAGS          ; Flags: window specifier.
        .DWORD  0
        .LOC    WIN_PKT+?WIN_PKT.PATH           ; Pathname of target window.
        .DWORD  0
        .LOC    WIN_PKT+?WIN_PKT.PATH_BUF_LEN   ; Length of pathname
        .WORD   0                               ; including null terminator.
        .LOC    WIN_PKT+?WIN_PKT.PATH_LEN       ; Length of pathname returned
        .WORD   0                               ; by AOS/VS.
        .LOC    WIN_PKT+?WIN_PKT.WIND_ID        ; Window ID of target window.
        .DWORD  0
        .LOC    WIN_PKT+?WIN_PKT.SUBPKT         ; Word pointer to function
        .DWORD  0                               ; subpacket or zero.
        .LOC    WIN_PKT+?WIN_PKT_LEN            ; End of packet.
```

Figure B–20. Listing of Program CREATE_WINDOW.SR (Continued)

```
        ; Set up storage for some ?WIN_CREATE parameters.

        PMAP:     .TXT    "@CONSOLE"
        PMAP_PATH_LENGTH = 9.

        WINDOW_NAME: .TXT "NEW_WINDOW"
        NAME_LEN = 11.

        WINDOW_TITLE: .TXT "CLI"
        TITLE_LEN = 4.

        WINDOW_PATH: .TXT "@CONSOLE:NEW_WINDOW"

        ; Define some masks for the ?WIN_CREATE flags word.

        NEW_GRP_MSK = 1S?WIN_CRE.FLAGS.NEW_GRP
        VT_SZ_X_MSK = 1S?WIN_CRE.FLAGS.VT_SZ_X
        VT_SZ_Y_MSK = 1S?WIN_CRE.FLAGS.VT_SZ_Y
        CRE_SON_MSK = 1S?WIN_CRE.FLAGS.CREATE_SON

        ; Set up the ?WIN_CREATE subpacket.

WIN_CRE: .BLK     ?WIN_CRE_LEN_REV1              ; Allocate enough space for
                                                 ; ?WIN_CREATE subpacket.
         .LOC     WIN_CRE+?WIN_CRE.PKT_ID        ; Packet ID.
         .DWORD   ?WIN_CRE_PKTID_REV1
         .LOC     WIN_CRE+?WIN_CRE.FLAGS         ; Flags: the window starts a new
                                                 ; group, and the width and
                                                 ; height of the virtual terminal
                                                 ; are specified with this call.
         .DWORD   NEW_GRP_MSK!VT_SZ_X_MSK!VT_SZ_Y_MSK
         .LOC     WIN_CRE+?WIN_CRE.VT_TYPE       ; Virtual terminal type:
         .DWORD   ?WIN_VT_TYPE_D460              ; D460 character window.
         .LOC     WIN_CRE+?WIN_CRE.VT_SZ.X       ; Width of virtual terminal is
         .DWORD   162.                           ; 162. (character) columns.
         .LOC     WIN_CRE+?WIN_CRE.VT_SZ.Y       ; Height of virtual terminal is
         .DWORD   24.                            ; 24. (character) rows.
         .LOC     WIN_CRE+?WIN_CRE.TITLE         ; Window title:
         .DWORD   WINDOW_TITLE*2                 ; byte pointer to WINDOW_TILLE.
         .LOC     WIN_CRE+?WIN_CRE.TITLE_BUF_LEN ; Length of window title
         .WORD    TITLE_LEN                      ; including a null terminator.
         .LOC     WIN_CRE+?WIN_CRE.PIXEL_DEPTH   ; Bits per pixel for pixel map:
         .WORD    0                              ; not valid for charcter window.
         .LOC     WIN_CRE+?WIN_CRE.INPUT_BUF_SZ  ; Length of input buffer is the
         .WORD    0                              ; default buffer size.
         .LOC     WIN_CRE+?WIN_CRE.WNAME_BUF_LEN ; Length of window name
         .WORD    name_len                       ; including a null terminator.
         .LOC     WIN_CRE+?WIN_CRE.WNAME         ; Window name:
         .DWORD   WINDOW_NAME*2                  ; byte pointer to WINDOW_NAME.
         .LOC     WIN_CRE+?WIN_CRE.PAL_TYPE      ; Palette type:
         .DWORD   ?WIN_PALETTE_TYPE_D460         ; use D460 palette.
         .LOC     WIN_CRE+?WIN_CRE.BORDER_TYPE   ; Border type:
         .WORD   ?WIN_BORDER_TYPE_TITLE          ; specify a title border type.
         .LOC     WIN_CRE+?WIN_CRE.RSV           ; Reserved value must be
         .WORD    0                              ; set to zero.
         .LOC     WIN_CRE+?WIN_CRE.VIEW_ORG.X    ; X coord. of view port origin:
         .DWORD   0                              ; set later with ?DEFINE_PORTS.
         .LOC     WIN_CRE+?WIN_CRE.VIEW_ORG.Y    ; Y coord. of view port origin:
         .DWORD   0                              ; set later with ?DEFINE_PORTS.
         .LOC     WIN_CRE+?WIN_CRE_LEN_REV1      ; End of subpacket.
```

Figure B-20. Listing of Program CREATE_WINDOW.SR (Continued)

Licensed Material - Property of Data General Corporation

```
        ; Set up the ?WIN_WINDOW_STATUS subpacket
WIN_STAT:.BLK   ?WIN_STATUS_LEN_REV1            ; Allocate enough space for
                                                ; ?WIN_WINDOW_STATUS subpacket.
        .LOC    WIN_STAT+?WIN_STATUS.PKT_ID     ; Packet ID.
        .DWORD  ?WIN_STATUS_PKTID_REV1
        .LOC    WIN_STAT+?WIN_STATUS.FLAGS      ; Window state flags.
        .DWORD  0
        .LOC    WIN_STAT+?WIN_STATUS.VIEW_ORG.X ; X-coordinate of view
        .DWORD  0                               ; port origin (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.VIEW_ORG.Y ; Y-coordinate of view
        .DWORD  0                               ; port origin (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.VIEW_SZ.X  ; Width of view port on
        .DWORD  0                               ; physical screen (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.VIEW_SZ.Y  ; Height of view port on
        .DWORD  0                               ; physical screen (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.SCAN_ORG.X ; X-coordinate of scan
        .DWORD  0                               ; port origin (chars).
        .LOC    WIN_STAT+?WIN_STATUS.SCAN_ORG.Y ; Y-coordinate of scan
        .DWORD  0                               ; port origin (chars).
        .LOC    WIN_STAT+?WIN_STATUS.SCAN_SZ.X  ; Width of scan port on
        .DWORD  0                               ; virtual screen (chars).
        .LOC    WIN_STAT+?WIN_STATUS.SCAN_SZ.Y  ; Height of scan port on
        .DWORD  0                               ; virtual screen (chars).
        .LOC    WIN_STAT+?WIN_STATUS.WIND_ORG.X ; X-coordinate of window
        .DWORD  0                               ; origin (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.WIND_ORG.Y ; Y-coordinate of window
        .DWORD  0                               ; origin (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.WIND_SZ.X  ; Width of window and
        .DWORD  0                               ; border (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.WIND_SZ.Y  ; Height of window and
        .DWORD  0                               ; border (pixels).
        .LOC    WIN_STAT+?WIN_STATUS.VT_SZ.X    ; Width of virtual
        .DWORD  0                               ; terminal (chars).
        .LOC    WIN_STAT+?WIN_STATUS.VT_SZ.Y    ; Height of virtual
        .DWORD  0                               ; terminal (chars).
        .LOC    WIN_STAT+?WIN_STATUS.VT_TYPE    ; Virtual terminal type.
        .DWORD  0
        .LOC    WIN_STAT+?WIN_STATUS.PIXEL_DEPTH ; Bits per pixel in
        .WORD   0                               ; pixel map.
        .LOC    WIN_STAT+?WIN_STATUS.INPUT_BUF_SZ ; Length of input
        .WORD   0                               ; buffer (bytes).
        .LOC    WIN_STAT+?WIN_STATUS.PAL_TYPE   ; Palette type.
        .DWORD  0
        .LOC    WIN_STAT+?WIN_STATUS_LEN_REV1   ; End of subpacket.
```

Figure B-20. Listing of Program CREATE_WINDOW.SR (Continued)

```
        ; Define some masks for the ?WIN_DEFINE_PORTS flag word.

        SCAN_SZ_X_MSK = 1S?WIN_DPORT.FLAGS.SCAN_SZ_X
        SCAN_SZ_Y_MSK = 1S?WIN_DPORT.FLAGS.SCAN_SZ_Y
        VIEW_ORG_X_MSK = 1S?WIN_DPORT.FLAGS.VIEW_ORG_X
        VIEW_ORG_Y_MSK = 1S?WIN_DPORT.FLAGS.VIEW_ORG_Y

        ; Set up the ?WIN_DEFINE_PORTS subpacket.

WIN_DPORT:  .BLK    ?WIN_DPORT_LEN              ; Allocate enough space for
                                                ; ?WIN_DEFINE_PORTS subpacket.
        .LOC    WIN_DPORT+?WIN_DPORT.PKT_ID     ; Packet ID.
        .DWORD  ?WIN_DPORT_PKTID

        .LOC    WIN_DPORT+?WIN_DPORT.FLAGS      ; Flags: this call specifies
                                                ; the dimensions of the scan
                                                ; port and the X coordinate of
                                                ; the view port's origin.
        .DWORD  SCAN_SZ_X_MSK!SCAN_SZ_Y_MSK!VIEW_ORG_X_MSK

        .LOC    WIN_DPORT+?WIN_DPORT.VIEW_ORG.X ; X-coordinate of view port
        .DWORD  100                             ; origin (pixels): 100.

        .LOC    WIN_DPORT+?WIN_DPORT.VIEW_ORG.Y ; Y-coordinate of view port
        .DWORD  0                               ; origin (pixels): not set.

        .LOC    WIN_DPORT+?WIN_DPORT.SCAN_ORG.X ; X-coordinate of scan port
        .DWORD  0                               ; origin (chars): not set.

        .LOC    WIN_DPORT+?WIN_DPORT.SCAN_ORG.Y ; Y-coordinate of scan port
        .DWORD  0                               ; origin (chars): not set.

        .LOC    WIN_DPORT+?WIN_DPORT.SCAN_SZ.X  ; Width of scan port (chars):
        .DWORD  80.                             ; 80. visible columns.

        .LOC    WIN_DPORT+?WIN_DPORT.SCAN_SZ.Y  ; Height of scan port (chars):
        .DWORD  24.                             ; 24. visible rows.

        .LOC    WIN_DPORT+?WIN_DPORT_LEN        ; End of subpacket.
```

Figure B–20. Listing of Program CREATE_WINDOW.SR (Continued)

Licensed Material - Property of Data General Corporation

```
        ; Set up the ?PROC packet
PROGRAM: .TXT    ":CLI.PR"                ; Program name of CLI to ?PROC
                                          ; in new window.

PROC_PKT: .BLK   ?PLTH                    ; Allocate enough space
                                          ; for ?PROC packet.

        .LOC     PROC_PKT+?PFLG           ; Process creation options: Block
        .WORD    0+?PFEX                  ; caller while son executes.

        .LOC     PROC_PKT+?PPRI           ; Priority of son process is
        .WORD    -1                       ; same as caller's priority.

        .LOC     PROC_PKT+?PSNM           ; Byte pointer to pathname of
        .DWORD   PROGRAM*2                ; program to exucute: ":CLI.PR"

        .LOC     PROC_PKT+?PIPC           ; Address of IPC message header.
        .DWORD   -1                       ; No initial IPC message.

        .LOC     PROC_PKT+?PNM            ; Simple process name of son is
        .DWORD   -1                       ; ASCII representation of PID.

        .LOC     PROC_PKT+?PMEM           ; Son has same logical page
        .DWORD   -1                       ; maximum as caller.

        .LOC     PROC_PKT+?PDIR           ; Son has same initial working
        .DWORD   -1                       ; directory as caller.

        .LOC     PROC_PKT+?PCON           ; Byte pointer to son's @CONSOLE
        .DWORD   WINDOW_PATH*2            ; file: set to window pathname.

        .LOC     PROC_PKT+?PCAL           ; Son is limited to two
        .WORD    -1                       ; concurrent system calls.

        .LOC     PROC_PKT+?PWSS           ; Son has no maximum working
        .WORD    -1                       ; set size.

        .LOC     PROC_PKT+?PUNM           ; Son has same username as
        .DWORD   -1                       ; caller.

        .LOC     PROC_PKT+?PPRV           ; Son has all caller's file
        .WORD    -1                       ; access privileges.

        .LOC     PROC_PKT+?PPCR           ; Son can create same maximum
        .WORD    -1                       ; number of sons as caller.

        .LOC     PROC_PKT+?PWMI           ; Son has no minimum working
        .WORD    -1                       ; set size.

        .LOC     PROC_PKT+?PIFP           ; Byte pointer to son's @INPUT
        .DWORD   WINDOW_PATH*2            ; file: set to window pathname.

        .LOC     PROC_PKT+?POFP           ; Byte pointer to son's @OUTPUT
        .DWORD   WINDOW_PATH*2            ; file: set to window pathname.

        .LOC     PROC_PKT+?PLFP           ; Son has same @LIST file as
        .DWORD   -1                       ; caller.

        .LOC     PROC_PKT+?PDFP           ; Son has same @DATA file as
        .DWORD   -1                       ; caller.

        .LOC     PROC_PKT+?SMCH           ; Son receives remainder of
        .DWORD   -1                       ; father's CPU time limit.

        .LOC    PROC_PKT+?PLTH            ; End of packet.

        .END    CREATE_WINDOW
```

Figure B-20. Listing of Program CREATE_WINDOW.SR (Concluded)

End of Appendix

Index

Within this index, "f" (or "ff") after a page number means "and the following page" (or "pages"). In addition, primary page references for each topic are listed first. Commands, calls, and acronyms are in uppercase letters (e.g., ?CREATE); all others are lowercase.

!·operator 1-4
16-bit process system calls (names of) 2-15
16-bit compatibility, ECLIPSE 1-30
16-bit compatible instructions, ECLIPSE 1-8
16-bit program counter, ECLIPSE 10-2f

A

Absolute addressing 1-11, 1-15
Absolute Value 11-132
Access and address translation, segment 9-2
AC0 1-2
AC1 1-2
AC2 1-2
AC3 1-2
accumulator(s) 1-2, 1-5
ACK0 character 2-662
ACK1 character 2-662
ADA 1-11
address, memory 1-1
?AEPR value 2-609
?ALAU value 2-609
?ALLOCATE system call 2-16f
?AOPR value 2-609
AOS/DVS system calls (names of) 2-4
AOS/VS system resources system calls (names of) 2-13
?APND value 2-383
?ASEB value 2-609
?ASID value 2-21
?ASLTH value 2-21
?ASPKT offset 2-21
?ASRSV offset 2-21
assembly language 1-2
  interface 1-6ff
  language programming 1-6ff
?ASSIGN system call 2-18
?ASSOC system call 2-19ff
?ASSUB offset 2-21
?ASVPID offset 2-21
?AUAL value 2-609
?AWENT value 2-24
?AWIRE system call 2-24
?AWUDS value 2-24

B

B operator 1-4
?BADL offset 2-26ff
?BADR offset 2-26ff
BASIC 1-11
?BBAC offset 2-26ff
?BBAL offset 2-26ff
?BBAN offset 2-26ff
?BBLC offset 2-26ff
?BBLL offset 2-26ff
?BBLN offset 2-26ff
BCC character 2-662
?BCHN offset 2-26ff
?BERR offset 2-26ff
bisynchronous communications system calls (names of) 2-14
bit
  mask 1-4
  position 1-4
  setting of a 1-4
?BITM value 2-355
bits
  least significant 1-5
  most significant 1-5
?BLBB offset 2-26ff
?BLKIO system call 2-25ff
?BLKPR system call 2-32f
?BLTH value 2-26
?BMBI value 2-125
?BMDA value 2-125
?BMDEV value 2-189
?BMFO value 2-126
?BMNO value 2-125
?BMNR value 2-125
?BMOP value 2-125
?BMPE value 2-125
?BMRA value 2-126

?BMSH value 2-125
?BMTI value 2-126
?BNAME system call 2-34
BOOMER.SR sample program B-33ff, B-2
?BPEL offset 2-26ff
?BPER offset 2-26ff
?BPVB offset 2-26ff
BRAC0 status word 2-36
BRAC1 status word 2-36
BRAC2 status word 2-36
BRAC3 status word 2-36
BRFP status word 2-36
bringing a window or group to the front 2-799
?BRKFL system call 2-35ff
BRPC status word 2-36
BRSB status word 2-36
BRSL status word 2-36
BRSP status word 2-36
BRTID status word 2-36
?BSTS offset 2-26ff
?BTBC offset 2-26ff
?BTBL offset 2-26ff
byte pointer 1-5

C

C 1-11
?C8BT value 2-175, 2-178, 2-645
?CACP offset 1-8, 2-70ff
call, system 1-1
?CCPS offset 2-74ff
?CDAY system call 2-38
?CDEH offset 2-74f
?CDEL offset 2-74ff
?CDRXON value 2-47f
?CDSBRK value 2-47f
?CDT0 value 2-176, 2-178, 2-646
?CDT1 value 2-178
?CDT2 value 2-178
?CDT3 value 2-176, 2-178, 2-646
?CEB0 value 2-176f, 2-645
?CEB1 value 2-176f, 2-645
?CEOC value 2-176f, 2-645
?CEOL value 2-175, 2-645
?CEOS value 2-176f, 2-645
?CEPI value 2-175, 2-645
?CESC value 2-176, 2-646
?CFF value 2-176, 2-645
?CFKT value 2-176, 2-646
?CFTYP 1-7f
?CFTYP offset 2-69ff, 2-72f, 2-74ff
?CGNAM system call 2-39
?CHAIN system call 2-40f
changing the view and scan ports 2-797ff
CHECK.F77 sample program B-45, B-2
?CHFS offset 1-7f, 2-72f
?CINT value 2-700, 2-714
?CKVOL system call 2-42
class scheduling system calls (names of) 2-13
?CLASS system call 2-43f
CLI 1-1
?CLMAX value 2-188, 2-667f
?CLMIN value 2-188, 2-667
?CLOSE system call 2-45f
  example of, *see* program WRITE.SR
closing a pixel map 2-221
?CLRDV system call 2-47f
?CLSCHED system call 2-49ff
?CLSTAT system call 2-53ff
?CLST_AOSVS value 2-56
?CLST_AOSVS.COREM offset 2-57f
?CLST_AOSVS.DAEMON offset 2-57f
?CLST_AOSVS.DISK offset 2-57f
?CLST_AOSVS.IDLE offset 2-57f
?CLST_AOSVS.INT offset 2-57f
?CLST_AOSVS.PKT_ID offset 2-57f
?CLST_AOSVS_PKTID value 2-57
?CLST_AOSVS.RESIDUE offset 2-57f
?CLST_AOSVS.SYSCALLS offset 2-57f
?CLST_AOSVS.SYSCALLS_MOM offset 2-57f
?CLST_AOSVS.SYSMGR offset 2-57f
?CLST_AOSVS.SYSTEM offset 2-57f
?CLST_NONE value 2-56
?CLST_PKT.CLASS offset 2-55f
?CLST_PKT.FUNC offset 2-54ff, 2-55f
?CLST_PKT.INTERVALS offset 2-55f
?CLST_PKT_LEN value 2-55
?CLST_PKT.LPID offset 2-55f
?CLST_PKT.PKT_ID offset 2-55f
?CLST_PKT_PKTID value 2-56
?CLST_PKT.SUBPKT offset 2-55f
?CLST_PKT.TOTAL offset 2-55f
?CLST_PKT.USER offset 2-55f
?CLST_PKT.USER_MOM offset 2-55f
?CLS_ACC value 2-51
?CLS_GET value 2-51
?CLS_PKT.FLAVOR offset 2-49, 2-51
?CLS_PKT.FUNC offset 2-51
?CLS_PKT.GETSET offset 2-49, 2-51
?CLS_PKT_LEN value 2-51
?CLS_PKT.LPMAP offset 2-49, 2-51
?CLS_PKT.PKT_ID offset 2-51
?CLS_PKT_PKTID value 2-51
?CLS_SCHED value 2-51
?CLS_SET value 2-51
?CLTH value 1-8, 2-70, 2-72, 2-74
?CL_GET value 2-44
?CL_PKT.CLMAP offset 2-44
?CL_PKT.FUNC offset 2-44
?CL_PKT.GETSET offset 2-44

Licensed Material - Property of Data General Corporation

?CL_PKT_LEN value 2-44
?CL_PKT.PKT_ID offset 2-44
?CL_PKT_PKTID value 2-44
?CL_SET value 2-44
?CMATRIX system call 2-59ff
?CMAT_CELL.CLASS offset 2-62f
?CMAT_CELL.LOC offset 2-62f
?CMAT_PKT.COUNT offset 2-61f
?CMAT_PKT.FUNC offset 2-61
?CMAT_PKT.GET value 2-61
?CMAT_PKT.GETSET offset 2-61
?CMAT_PKT_LEN value 2-61
?CMAT_PKT.MATRIX offset 2-61, 2-63
?CMAT_PKT.PID_ID offset 2-61
?CMAT_PKT.PKT_ID offset 2-63
?CMAT_PKT_PKTID value 2-61
?CMAT_PKT.SET value 2-61
?CMIL 1-8
?CMIL offset 2-72f, 2-74ff
?CMOD value 2-176, 2-178, 2-644, 2-646, 2-668
?CMPLT value 2-67
?CMRI value 2-175, 2-178, 2-644f, 2-668
?CMRS offset 1-8, 2-72f, 2-74
?CMSH offset 1-8, 2-72f
?CNAS value 2-175, 2-177, 2-645
?CNNL value 2-176, 2-178, 2-646
?CNRM value 2-176, 2-646, 2-670
COBOL 1-11
CODE MASM macro 2-114
code(s), error 1-1f
  listing of A-1ff
coding, parametric 1-3
?CON system call 2-64f
connection management system calls (names of) 2-12
construction, program 1-9
?CONT value 2-700, 2-706, 2-714
controlling
  keyboard input to a window 2-801
  pointer appearance 2-573f
  window output 2-801
  window priority 2-799
  window visibility 2-800
?COTT value 2-175, 2-177, 2-645
?CPBN value 2-176, 2-178, 2-646
?CPM value 2-176, 2-178, 2-646
?CPMAX system call 2-66f
?CPMCN offset 2-67
?CPMFW offset 2-67
?CPMHS offset 2-67
?CPMLS offset 2-67
?CPOR offset 2-70f
?CRAC value 2-175, 2-645
?CRAF value 2-175, 2-645
?CRAT value 2-175, 2-645
?CREATE system call 2-68ff, 1-1, 1-6ff, 1-12
  example of, *see* program DIRCREATE.F77
  example of, *see* program FILCREATE.SR
  example of, *see* program HEAR.SR
  example of, *see* program SPEAK.SR
CREATE_WINDOW.SR sample program B-49ff, B-2
creating
  a new window 2-792ff
  a pixel map in memory 2-219ff
?CRT1 value 2-646
?CRT2 value 2-646
?CRT3 value 2-176, 2-178, 2-646
?CRT6 value 2-176, 2-178, 2-646
?CRT7 value 2-646
?CRT8 value 2-646
?CRT9 value 2-646
?CRT10 value 2-646
?CRT11 value 2-646
?CRT12 value 2-646
?CRT13 value 2-646
?CRT14 value 2-646
?CRT15 value 2-646
?CRUDA system call 2-78f
?CSBDS value 2-668
?CSBEN value 2-668
?CSFF value 2-175, 2-645
?CSPO value 2-175, 2-645
?CST value 2-175, 2-645
?CTERM system call 2-80f
?CTIM offset 1-8, 2-70f, 2-72f, 2-74ff
?CTO value 2-176, 2-646
?CTOD system call 2-82
?CTSP value 2-176, 2-178, 2-646
?CTYPE system call 2-83f
?CUCO value 2-175, 2-645
?CULC value 2-176f, 2-646
?CWIN value 2-189
?CWRP value 2-176, 2-646

D

?DAC2 offset 2-103, 2-737f
?DACL system call 2-85f
?DADID system call 2-87
?DCC offset 2-103, 2-741f
?DCI offset 2-103, 2-741f
?DCL2 offset 2-737
?DCON system call 2-88f
?DDIS system call 2-90
?DEASSIGN system call 2-91
?DEBL system call 2-92
?DEBUG system call 2-93
  example of, *see* program INRING.SR
debugging system calls (names of) 2-8f

?DELAY system call 2-94
?DELETE system call 2-95f
  example of, *see* program FILCREATE.SR
deleting a window 2-796
delimiter table 2-389
?DEREGISTER system call 2-97f
device code 10 note 2-595
device code 11 note 2-595
?DFLGS offset 2-103, 2-737f
?DFLRC value 2-738
?DFRSCH system call 2-99
DG/VIEW windowing user interface 2-802
?DID offset 2-103, 2-737f
?DIID value 2-22
?DILTH value 2-22
?DIPKT offset 2-22
?DIR 1-6ff, 1-12
?DIR system call 2-100, 1-6ff, 1-12
  example of, *see* program DIRCREATE.F77
  example of, *see* program DIRCREATE.SR
DIRCREATE.F77 sample program B-42f, B-2
DIRCREATE.SR sample program 1-6ff
?DIRSV offset 2-22
?DISSOC system call 2-19ff, 2-101
?DISUB offset 2-22
?DIVPID offset 2-22
DLE character 2-662
DLE EOT characters 2-662
DLIST.SR sample program B-27ff, B-2
?DLNK offset 2-103, 2-737f
?DLNKB offset 2-103, 2-737f
?DLNKBL offset 2-737
?DLNKL offset 2-103
?DLNL offset 2-737
?DNUM offset 2-103, 2-737f
?DPC offset 2-103, 2-737f
?DPCL offset 2-737
?DPRI offset 2-103, 2-737f
?DQTSK system call 2-102
?DRCON system call 2-104
?DRES offset 2-103, 2-737f
?DRSCH system call 2-105
?DSCH value 2-99
?DSFLT offset 2-103, 2-737f
?DSH offset 2-103, 2-741f
?DSLTH value 2-737
?DSMS offset 2-103, 2-741f
DSR value 2-666
?DSSL offset 2-737f
?DSSZ offset 2-103, 2-737f
?DSTB offset 2-103, 2-737f
?DSTL offset 2-737
.DWORD MASM pseudo-operation 1-5
?DXLTH value 2-103, 2-741

E

?EBAS value 2-609
?EFFP offset 2-596, 2-608f
?EFLN offset 2-596, 2-608f
?EFMAX value 2-608
?EFNF offset 2-596, 2-608f
?EFTL offset 2-608f
?EFTY offset 2-596, 2-608f
?ELAC offset 2-393, 2-395
?ELCR offset 2-393f
?ELCT offset 2-393f
?ELFS offset 2-393, 2-395
?ELGN offset 2-393f
?ELL1 value 2-395
?ELL2 value 2-395
?ELL3 value 2-395
?ELLN value 2-393
?ELRE offset 2-393f
?ELUH offset 2-393, 2-395
?ELUT offset 2-393, 2-395
?ELVL offset 2-393f
?ELVR offset 2-393f
?ENBFL offset 2-108
?ENBLN value 2-108
?ENBRK system call 2-106ff
?ENCST value 2-108
?ENDIR value 2-108f
?ENESH offset 2-108f
?ENET offset 2-46, 2-382, 2-386, 2-596
?ENEUS offset 2-108f
?ENFNP offset 2-108f
?ENOV value 2-588
?ENPRE value 2-108
ENQ character 2-663
?ENQUE system call 2-110f
?ENR4 value 2-108
?ENR5 value 2-108
?ENR6 value 2-108
?ENR7 value 2-108
?ENSH value 2-108
?ENSSH offset 2-108f
?ENSUS offset 2-108
?ENUS value 2-108
EOT character 2-663
?ERBA offset 2-672
?ERCH offset 2-672
?ERCS offset 2-672
?ERFT offset 2-46
?ERLTH value 2-672
ERMES file 2-112ff

Licensed Material - Property of Data General Corporation

?ERMSG system call 2-112ff
error
codes 1-1f, A-1ff
reporting 1-10
return 1-2
?ERSCH system call 2-115
?ESBE value 2-604
?ESCP value 2-604
?ESCR offset 2-596, 2-603f
?ESDD value 2-604
?ESED value 2-604
?ESEP offset 2-596, 2-603f
?ESFC offset 2-596, 2-603f
?ESFF system call 2-116
?ESGT value 2-604
?ESLN value 2-603
?ESNE value 2-604
?ESNR value 2-604
?ESPE value 2-604
?ESRD value 2-604
?ESRP value 2-604
?ESSE value 2-604
ETB character 2-663
?ETBB value 2-706, 2-713
?ETER offset 2-46, 2-382, 2-386
?ETFL offset 2-596
?ETFT offset 2-210, 2-382, 2-596
?ETLL offset 2-596
?ETLT offset 2-46, 2-210, 2-382, 2-386, 2-596
?ETMX value 2-596
?ETSL offset 2-596
?ETSP offset 2-46, 2-210, 2-382, 2-596
ETX character 2-663
?ETXB value 2-706, 2-713
?EXAC value 2-773
exception codes 1-2
?EXEC functions
changing queuing parameters 2-140
dismounting a unit (extended request) 2-139
dismounting unlabeled and labeled tapes 2-121
holding, unholding, canceling queue requests (AOS/DVS) 2-133
holding, unholding, canceling queue requests (AOS/VS) 2-131
mounting unlabeled and labeled tapes 2-118ff
obtaining EXEC status information 2-132
obtaining extended EXEC status information 2-136
obtaining QDISPLAY information 2-145
obtaining queue names 2-143
queuing a file entry 2-122
submitting a job to a MOUNT queue 2-138
?EXEC system call 2-117ff
execution, program 1-9
?EXPO system call 2-148f
?EXTG pseudo-operation B-15

F

?FAAB value 2-235f
?FACA value 2-86, 2-198, 2-236, 2-513
?FACE value 2-86, 2-198, 2-236, 2-513
?FACO value 2-86, 2-198, 2-236, 2-513
?FACR value 2-86, 2-198, 2-236, 2-513
?FACW value 2-86, 2-198, 2-236, 2-513
?FAEA value 2-166
?FAEB value 2-235f
?FAOB value 2-235f
?FARA value 2-166
?FARB value 2-235f
?FAWB value 2-235f
?FBEX offset 2-153, 2-155
?FBSTF offset 2-153
?FCPC offset 2-153ff
?FCPD file type 2-69, 2-72f, 2-74f, 2-388
?FDAY system call 2-150
?FDBA offset 2-153, 2-155
?FDBFZ offset 2-154
?FDBL offset 2-153, 2-155
?FDIR 1-7f
?FDIR file type 2-69, 2-72f, 2-74f, 2-388
?FDLE value 2-166
?FEDFUNC system call 2-151ff
?FEOV system call 2-156
?FEXPR value 2-155
?FFCC file type 2-69
?FFLAG offset 2-152ff
?FFLPT value 2-155
?FGLT file type 2-69
FILCREATE.SR sample program B-20f, B-2
file creation and management system calls (names of) 2-7
file input/output system calls (names of) 2-8f
?FINA offset 2-154
?FINST value 2-155
?FIPC file type 2-69ff, 2-74f, 2-388
?FIPC value 2-199
?FLCC file type 2-69
?FLCHN offset 2-158, 2-170
?FLCR value 2-152
?FLDIS value 2-154
?FLEFS value 2-153
?FLEX offset 2-153, 2-155
?FLLEN value 2-158, 2-170
?FLNK file type 2-69
?FLOCK system call 2-157
?FLOG file type 2-732
?FLOST value 2-153

?FLPID offset 2-158, 2-170
?FLREV offset 2-158, 2-170
?FLRSW offset 2-158, 2-170
?FLSEL offset 2-158, 2-170
?FLSYM value 2-154f
?FLTY offset 2-158, 2-170
?FLUSH system call 2-159
?FMDB value 2-166
?FMEFS value 2-155
?FNCC file type 2-69
?FNIR offset 2-152
?FOCC file type 2-69
format of time value 2-54
FORTRAN 77 1-11f
?FPIP file type 2-69
?FPRG file type 2-69
?FPRM value 2-166
?FPRV file type 2-69, 2-388
?FQUE file type 2-69
frame pointer 1-2
?FRCR code 2-152
?FRDIS code 2-152, 2-154
?FRDTS code 2-152, 2-155
?FREFS code 2-153, 2-152, 2-155
?FRESD offset 2-153, 2-155
?FRESS offset 2-152
?FRFNC offset 2-152ff
?FRITS code 2-152, 2-154
?FROST code 2-153, 2-152
?FRRR offset 2-154
?FSDF file type 2-69
?FSHB value 2-166
?FSNL offset 2-154f
?FSNM offset 2-154f
?FSPR file type 2-69ff, 2-74f
?FSTAT system call 2-160ff
?FSTF file type 2-69
?FSVAL offset 2-154f
?FSVLL offset 2-154f
?FTCK value 2-158
?FTER value 2-158
?FTEX value 2-158
?FTOD system call 2-168
?FTPN value 2-158
?FTSH value 2-158
?FTXT file type 2-69, 2-388
?FUDA value 2-166
?FUDF file type 2-69, 2-385, 2-388
?FULA value 2-170
?FUNLOCK system call 2-169f
?FUNX file type 2-69
?FUPF file type 2-69
?FWFI offset 2-154
?FWFL value 2-158

G

?GACL system call 2-171f
?GARG value 2-240, 2-242
GATE.ARRAY.SR sample program B-18, B-2
?GBIAS system call 2-173
?GCHR system call 2-174ff
?GCLOSE system call 2-180f
?GCMD value 2-240, 2-242
?GCNT value 2-240, 2-242
?GCPCN offset 2-79, 2-96, 2-172, 2-620, 2-638, 2-640
?GCPFW offset 2-79, 2-96, 2-172, 2-620, 2-638, 2-640
?GCPLT value 2-79, 2-96, 2-172, 2-620, 2-638, 2-640
?GCPN system call 2-182f
?GCRB system call 2-184
?GDAY system call 2-185
?GDLM system call 2-186f
?GECHR system call 2-188f
generating a pointer event 2-577ff
getting
  a window's status 2-808ff
  a window's title 2-806f
  information about the last pointer event 2-571ff
  the current windowing user interface settings 2-804ff
  the pointer status 2-575f
  the status of a physical device 2-813ff
  the status of a pixel map 2-221ff
  window IDs for windows in a group 2-815
  window IDs for windows on a physical terminal 2-816
  window IDs 2-813ff
?GFCF value 2-240
?GHRZ system call 2-190f
?GLINK system call 2-191
?GLIST system call 2-192
?GMEM system call 2-193
?GMES value 2-240, 2-242
?GNAME system call 2-194
?GNFN 1-1
?GNFN system call 2-195ff
  example of, *see* program DLIST.SR
?GNUM offset 2-238f, 2-241
?GOPEN system call 2-198ff
  example of, *see* program DLIST.SR
?GPID system call 2-205ff
?GPORT system call 2-208
?GPOS system call 2-209ff
?GPRNM system call 2-212
?GRAPHICS system call 2-213ff
?GRAPH_CLOSE_PIXELMAP function code 2-221, 2-214

Licensed Material - Property of Data General Corporation

?GRAPH_CREATE_MEMORY_PIXELMAP function code 2-219ff, 2-214, 2-216
?GRAPH_CRE.FLAGS offset 2-219f
?GRAPH_CRE.FLAGS.PIX_DEPTH value 2-220
?GRAPH_CRE.FLAGS.SZ_X value 2-220
?GRAPH_CRE.FLAGS.SZ_Y value 2-220
?GRAPH_CRE_LEN value 2-219
?GRAPH_CRE.PIXEL_DEPTH offset 2-219f
?GRAPH_CRE.PIXMAP_SZ.X offset 2-219f
?GRAPH_CRE.PIXMAP_SZ.Y offset 2-219f
?GRAPH_CRE.PKT_ID offset 2-219f
?GRAPH_CRE_PKTID value 2-220
?GRAPH_MAP_PIXELMAP function code 2-227ff, 2-214
?GRAPH_MAP_PMAP.ADDR offset 2-227f
?GRAPH_MAP_PMAP.FORMAT offset 2-227f
?GRAPH_MAP_PMAP.FORMAT.PACKED_PIXEL value 2-228
?GRAPH_MAP_PMAP_LEN value 2-227
?GRAPH_MAP_PMAP.PKT_ID offset 2-227f
?GRAPH_MAP_PMAP_PKTID value 2-228
?GRAPH_MAP_PMAP.X_PITCH offset 2-227f
?GRAPH_MAP_PMAP.Y_PITCH offset 2-227f
?GRAPH_OPEN.CHAN_NUM offset 2-217f
?GRAPH_OPEN.FLAGS offset 2-217f
?GRAPH_OPEN.FLAGS value 2-217
?GRAPH_OPEN.FLAGS.IN_CHAN value 2-218
?GRAPH_OPEN.FLAGS.IN_PATH value 2-218
?GRAPH_OPEN.FLAGS.IN_WIND_ID value 2-218
?GRAPH_OPEN.FLAGS.READ_ONLY value 2-218
?GRAPH_OPEN_LEN value 2-217
?GRAPH_OPEN.PATH offset 2-217f
?GRAPH_OPEN.PATH_BUF_LEN offset 2-217f
?GRAPH_OPEN.PATH_LEN offset 2-217f
?GRAPH_OPEN.PKT_ID offset 2-217f
?GRAPH_OPEN_PKTID value 2-218
?GRAPH_OPEN.RSV offset 2-217f
?GRAPH_OPEN.WIND_ID offset 2-217f
?GRAPH_OPEN_WINDOW_PIXELMAP function code 2-216ff, 2-214
?GRAPH_PIXELMAP_STATUS function code 2-221ff, 2-214
?GRAPH_PIXSTAT.CLIP_ORG.X offset 2-222f
?GRAPH_PIXSTAT.CLIP_ORG.Y offset 2-222f
?GRAPH_PIXSTAT.CLIP_STATE offset 2-222f
?GRAPH_PIXSTAT.CLIP_SZ.X offset 2-222f
?GRAPH_PIXSTAT.CLIP_SZ.Y offset 2-222f
?GRAPH_PIXSTAT_LEN value 2-222
?GRAPH_PIXSTAT.PIXEL_DEPTH offset 2-222f
?GRAPH_PIXSTAT.PIXMAP_SZ.X offset 2-222f
?GRAPH_PIXSTAT.PIXMAP_SZ.Y offset 2-222f
?GRAPH_PIXSTAT.PKT_ID offset 2-222f
?GRAPH_PIXSTAT_PKTID value 2-223
?GRAPH_PIXSTAT.RSV offset 2-222f
?GRAPH_PKT.FUNC offset 2-215f
?GRAPH_PKT_LEN value 2-215
?GRAPH_PKT.PIXMAP_ID offset 2-215f, 2-216, 2-219
?GRAPH_PKT.PKT_ID offset 2-215f
?GRAPH_PKT_PKTID value 2-216
?GRAPH_PKT.RSV offset 2-215f
?GRAPH_PKT.SUBPKT offset 2-215f, 2-221, 2-229
?GRAPH_RDPAL.ARRAY_PKT_ID offset 2-231
?GRAPH_RDPAL_ARRAY_PKTID value 2-231
?GRAPH_RDPAL.ENTRY_CNT offset 2-231
?GRAPH_RDPAL.FIRST_PAL_INDEX offset 2-231
?GRAPH_RDPAL_LEN value 2-231
?GRAPH_RDPAL.PH0_ARRAY offset 2-231
?GRAPH_RDPAL.PH1_ARRAY offset 2-231
?GRAPH_RDPAL.PKT_ID offset 2-231
?GRAPH_RDPAL_PKTID value 2-231
?GRAPH_READ_PALETTE function code 2-230f, 2-214
?GRAPH_RECT_STATE_DISABLE value 2-223, 2-226
?GRAPH_RECT_STATE_ENABLE value 2-223, 2-226
?GRAPH_SET_CLIP.FLAGS offset 2-224ff
?GRAPH_SET_CLIP.FLAGS.ORG_X value 2-225
?GRAPH_SET_CLIP.FLAGS.ORG_Y value 2-225
?GRAPH_SET_CLIP.FLAGS.STATE value 2-225
?GRAPH_SET_CLIP.FLAGS.SZ_X value 2-225
?GRAPH_SET_CLIP.FLAGS.SZ_Y value 2-225
?GRAPH_SET_CLIP.LEN value 2-224
?GRAPH_SET_CLIP.ORG.X offset 2-224ff
?GRAPH_SET_CLIP.ORG.Y offset 2-224ff
?GRAPH_SET_CLIP.PKT_ID offset 2-224ff
?GRAPH_SET_CLIP_PKTID value 2-225
?GRAPH_SET_CLIP_RECTANGLE function code 2-223ff, 2-214
?GRAPH_SET_CLIP.RSV offset 2-224ff
?GRAPH_SET_CLIP.STATE offset 2-224ff
?GRAPH_SET_CLIP.SZ.X offset 2-224ff
?GRAPH_SET_CLIP.SZ.Y offset 2-224ff
?GRAPH_UNMAP_PIXELMAP function code 2-229ff, 2-214
?GRAPH_WRITE_PALETTE function code 2-229f, 2-214
?GRAPH_WRPAL.ARRAY_PKT_ID offset 2-230
?GRAPH_WRPAL_ARRAY_PKTID value 2-230
?GRAPH_WRPAL.ENTRY_CNT offset 2-230
?GRAPH_WRPAL.FIRST_PAL_INDEX offset 2-230
?GRAPH_WRPAL_LEN value 2-230
?GRAPH_WRPAL.PH0_ARRAY offset 2-230
?GRAPH_WRPAL.PH1_ARRAY offset 2-230
?GRAPH_WRPAL.PKT_ID offset 2-230
?GRAPH_WRPAL_PKTID value 2-230

?GRCH offset 2-634
?GREQ offset 2-238ff
?GRES offset 2-238ff
?GRIH offset 2-634
?GRLTH value 2-634
?GRNAME system call 2-232f
GRP MASM macro 2-114
?GRPH offset 2-634
?GRRH offset 2-634
?GSHPT system call 2-233
?GSID system call 2-234
?GSW offset 2-238ff
?GSWS value 2-241f
?GTACP system call 2-235f
?GTMES system call 2-237f
  example of, *see* program BOOMER.SR
  example of, *see* program DLIST.SR
  example of, *see* program TIMEOUT.SR
?GTNAM system call 2-245f
?GTOD system call 2-247
?GTRUNCATE system call 2-248f
?GTSVL system call 2-250f
?GTSW value 2-240ff
?GUNM system call 2-252f
?GVPID system call 2-254

H

?HAPH array offset 2-274, 2-785
?HAPL array offset 2-274, 2-785
?HARAY array offset 2-274, 2-785
HEAR.SR sample program B-3ff, B-2
?HIBUF offset 2-273f
?HIEND offset 2-273f
high-level language 1-2
high-level language interface 1-11, 1-2, 1-12
high-order bits 1-5
?HIST offset 2-273f
?HIWDS offset 2-273f
?HNAME system call 2-255f
?HPRH array offset 2-274, 2-785
?HPRL array offset 2-274, 2-785
?HSBH array offset 2-274, 2-785
?HSBL array offset 2-274, 2-785
?HSIH array offset 2-274, 2-785
?HSIL array offset 2-274, 2-785
?HTTH array offset 2-274, 2-785
?HTTL array offset 2-274, 2-785
?HWBUF offset 2-784
?HWEND offset 2-784
?HWLTH value 2-273, 2-784
?HWST offset 2-784
?HWWDS offset 2-784

I

?IBAD offset 2-46, 2-210, 2-382, 2-385, 2-596, 2-599
?IBIN value 2-598
?IBLT value 2-210, 2-382
?ICH offset 2-46, 2-210, 2-382f, 2-596, 2-598
?ICRF value 2-383, 2-598
?ID16 value 2-385
?ID62 value 2-385
?ID8 value 2-385
?IDAM value 2-385
?IDEF system call 2-257ff
?IDEL offset 2-46, 2-210, 2-382, 2-386, 2-389, 2-596, 2-600
identifier, unique task 2-677, 2-679, 2-768f
?IDGOTO system call 2-264
?IDKIL system call 2-265
  example of, *see* program NEWTASK.SR
?IDPH offset 2-294f, 2-298f
?IDPN offset 2-283f
?IDPRI system call 2-266
?IDRDY system call 2-267
?IDSTAT system call 2-268
?IDSUS system call 2-269
?IESS system call 2-270
?IEXO value 1-4, 2-383
?IFBNK value 2-299
?IFNBK value 2-284
?IFNP offset 2-46, 2-210, 2-382, 2-386, 2-596, 2-600
?IFNSP value 2-294, 2-299
?IFOP value 2-599
?IFPR value 2-284, 2-299
?IFPU system call 2-271
?IFRFM value 2-284, 2-299
?IFRING value 2-284, 2-299
?IFSOV value 2-284, 2-299
?IFSTM value 2-294, 2-299
?IHIST system call 2-272ff
?ILKUP system call 2-275
  example of, *see* program HEAR.SR
  example of, *see* program SPEAK.SR
?ILTH offset 2-283, 2-294f, 2-298f
?ILTH value 2-284
?IMERGE system call 2-276
?IMFF value 2-384
?IMIO value 2-588
?IMNH value 2-384
?IMP2 value 2-384
?IMRS offset 2-46, 2-76, 2-210, 2-382, 2-385, 2-596, 2-599
?IMSG system call 2-277
?INID value 2-412
?INIT system call 2-278f, 2-828
INRING.SR sample program B-17f, B-2

Licensed Material - Property of Data General Corporation

interface
  assembly language 1-6ff
  high-level language 1-2
interprocess communications system calls (names of) 2-11
?INTWT system call 2-280
?IOPH offset 2-283f
?IOPN offset 2-294, 2-298f
?IOSZ value 2-46, 2-382, 2-596
?IPKL value 2-383, 2-598
?IPLTH value 2-283, 2-294
?IPRLTH value 2-298
?IPST offset 2-211, 2-600
?IPST value 2-598, 2-696f
?IPTR offset 2-283, 2-286f, 2-294f, 2-298f
?IPTR value 2-284
?IQTSK system call 2-281
?IRCL offset 2-46, 2-210, 2-382, 2-385, 2-596, 2-599, 2-697
?IRCL value 2-696
?IREC system call 2-282ff
  example of, *see* program HEAR.SR
?IRES offset 2-46, 2-210, 2-382, 2-385, 2-596, 2-599
?IRLR offset 2-46, 2-210, 2-382, 2-386, 2-596, 2-600
?IRLT offset 2-298f
?IRMV system call 2-292
?IRNH offset 2-46, 2-210f, 2-382, 2-386, 2-596, 2-600
?IRNH value 2-696f
?IRNW offset 2-46, 2-210, 2-382, 2-386, 2-596, 2-600
?IRPT offset 2-298f
?IRSV offset 2-298f
IS.R system call 2-297ff
?ISEND system call 2-293ff
  example of, *see* program SPEAK.SR
?ISFL offset 2-283f, 2-294, 2-298f
?ISPLIT system call 2-296
?ISTI offset 1-4, 2-46, 2-210, 2-382ff, 2-387, 2-596, 2-598f, 2-696
?ISTO offset 2-46, 2-210, 2-382, 2-384f, 2-596, 2-599
ISYS FORTRAN 77 function 1-12
ITB character 2-664
?ITIME system call 2-301
?IUFL offset 2-285, 2-81, 2-283f, 2-287, 2-289, 2-294, 2-298f
?IXIT system call 2-302
?IXMT system call 2-303

J

?JPID_MAX value 2-306, 2-308, 2-310
?JPID_MIN value 2-306, 2-308, 2-310
?JPINIT system call 2-304ff
?JPI_PKT.FLAGS offset 2-305f
?JPI_PKT.FLAGS_EX_UC offset 2-305f
?JPI_PKT.FLAGS_IN_UC offset 2-305f
?JPI_PKT.FUNC offset 2-305f
?JPI_PKT.JPID offset 2-305f
?JPI_PKT_LEN value 2-305
?JPI_PKT.LPID offset 2-305f
?JPI_PKT.PKT_ID offset 2-305f
?JPI_PKT_PKTID value 2-306
?JPI_PKT.UC_NAME offset 2-305f
?JPI_PKT.UC_OPTS offset 2-305f
?JPI_PKT.UC_SZ offset 2-305f
JPLCS instruction 2-306
?JPMOV system call 2-307f
?JPM_PKT.FLAGS offset 2-308
?JPM_PKT.FLAGS_LJPID value 2-308
?JPM_PKT.FUNC offset 2-308
?JPM_PKT.JPID offset 2-308
?JPM_PKT_LEN value 2-308
?JPM_PKT.LPID offset 2-308
?JPM_PKT.PKT_ID offset 2-308
?JPM_PKT_PKTID value 2-308
?JPREL system call 2-309f
?JPR_PKT.FLAGS offset 2-310
?JPR_PKT.FLAGS_LJPID value 2-310
?JPR_PKT.FUNC offset 2-310
?JPR_PKT.JPID offset 2-310
?JPR_PKT_LEN value 2-310
?JPR_PKT.PKT_ID offset 2-310
?JPR_PKT_PKTID value 2-310
?JPSTAT system call 2-311ff
?JPS_GEN value 2-312
?JPS_GEN.JPCNT offset 2-313
?JPS_GEN.JPMAP offset 2-313
?JPS_GEN.PKT_ID offset 2-313
?JPS_GEN_PKTID value 2-313
?JPS_GEN_LEN value 2-313
?JPS_PKT.FUNC offset 2-312
?JPS_PKT_LEN value 2-312
?JPS_PKT.PKT_ID offset 2-312
?JPS_PKT_PKTID value 2-312
?JPS_PKT.SUBPKT offset 2-312
?JPS_SPEC value 2-312
?JPS_SPEC.FLAGS offset 2-314
?JPS_SPEC.FLAGS_MOM value 2-314
?JPS_SPEC.FLAGS_SYSTSK value 2-314
?JPS_SPEC.JPID offset 2-314

?JPS_SPEC_LEN value 2-314
?JPS_SPEC.LPID offset 2-314
?JPS_SPEC.MODEL offset 2-314
?JPS_SPEC.PKT_ID offset 2-314
?JPS_SPEC_PKTID value 2-314
?JPS_SPEC.STATE offset 2-314
?JPS_SPEC.STATUS offset 2-314
?JPS_SPEC.UC_REV offset 2-314

K

?KCALL system call 2-315
?KHIST system call 2-316
?KILAD system call 2-317
?KILL system call 2-318
  example of, see program BOOMER.SR
?KINTR system call 2-319
?KIOFF system call 2-320
?KION system call 2-321
?KWAIT system call 2-322

L

?LABEL system call 2-323ff
language
  assembly 1-2
  high-level 1-2
  interface, high-level 1-11, 1-2, 1-12
?LB8 value 2-324f
?LB16 value 2-324f
?LB62 value 2-324f
?LBAC offset 2-324f
?LBAM value 2-325
?LBDV offset 2-324f
?LBFG offset 2-324f
?LBIM value 2-325
?LBLN value 2-324
?LBOI offset 2-324f
?LBSC value 2-325
?LBST offset 2-324f
?LBUV offset 2-324f
?LBVD offset 2-324f
LCALL instruction B-15
?LDMA code 2-339
least significant bits 1-5
?LEFD system call 2-327
?LEFE system call 2-328
?LEFS system call 2-329
?LFOP value 2-335
Link 1-9
?LMAP system call 2-330f
?LMAX code 2-339
?LOCALITY system call 2-332ff
?LOC_PID value 2-334
?LOC_PKT.FUNC offset 2-333f
?LOC_PKT_LEN value 2-333
?LOC_PKT.PCODE offset 2-333f
?LOC_PKT.PID offset 2-333f
?LOC_PKT.PKT_ID offset 2-333f
?LOC_PKT_PKTID value 2-334
?LOC_PKT.PNAME offset 2-333f
?LOC_PKT.STRLEN offset 2-333f
?LOC_PKT.ULOC offset 2-333f
?LOC_PNAME value 2-334
log file, system 2-338f
?LOGCALLS system call 2-335ff
?LOGEV system call 2-338f
?LOGHREC value 2-337
low-order bits 1-5
?LPCLASS system call 2-340ff
?LPCL_PKT.BASE offset 2-341f
?LPCL_PKT.FUNC offset 2-341f
?LPCL_PKT.GETSET offset 2-341f
?LPCL_PKT_LEN value 2-341
?LPCL_PKT.LPID offset 2-341f
?LPCL_PKT.PKT_ID offset 2-341f
?LPCL_PKT_PKTID value 2-342
?LPCL_PKT.TIME offset 2-341f
?LPCL_PKT.TIME_DEF value 2-340
?LPCL_PKT.TIME_MAX value 2-340
?LPCL_PKT.TIME_MIN value 2-340
?LPCREA system call 2-343f
?LPC_PKT.FUNC offset 2-344
?LPC_PKT_LEN value 2-344
?LPC_PKT.LPID offset 2-344
?LPC_PKT.PKT_ID offset 2-344
?LPC_PKT_PKTID value 2-344
?LPDELE system call 2-345f
?LPD_PKT.FUNC offset 2-346
?LPD_PKT_LEN value 2-346
?LPD_PKT.LPID offset 2-346
?LPD_PKT.PKT_ID offset 2-346
?LPD_PKT_PKTID value 2-346
?LPID_MAX value 2-306, 2-308, 2-343
?LPID_MIN value 2-306, 2-308, 2-343
?LPSTAT system call 2-347ff
?LPS_FUNC_MAX value 2-349
?LPS_FUNC_MIN value 2-349
?LPS_GEN value 2-349
?LPS_GEN_LEN value 2-350
?LPS_GEN.LPCNT offset 2-350
?LPS_GEN.LPMAP offset 2-350
?LPS_GEN.PKT_ID offset 2-350
?LPS_GEN_PKTID value 2-350
?LPS_PKT.FUNC offset 2-349
?LPS_PKT_LEN value 2-349
?LPS_PKT.PKT_ID offset 2-349
?LPS_PKT_PKTID value 2-349
?LPS_PKT.SUBPKT offset 2-349

Licensed Material - Property of Data General Corporation

?LPS_SPEC value 2-349
?LPS_SPEC.JPCNT offset 2-351
?LPS_SPEC.JPMAP offset 2-351
?LPS_SPEC_LEN value 2-351
?LPS_SPEC.LPID offset 2-351
?LPS_SPEC.PKT_ID offset 2-351
?LPS_SPEC_PKTID value 2-351
?LSMI code 2-339
?LSTART value 2-335
?LUMI code 2-339

M

macroassembler 1-9
making a hidden window or group visible 2-800
making a window or group invisible 2-800
?MAPDV system call 2-352ff
?MAPDV_PKT_PKTID value 2-355
mapping a pixel map into your program's address space 2-227ff
mask, bit 1-4
MASM.PR 1-9
MASM.PS 1-9
?MBAH offset 2-357, 2-359
?MBBC offset 2-357, 2-359
?MBCH offset 2-357, 2-359
?MBFC system call 2-356f
?MBID offset 2-357, 2-359
?MBLTH value 2-357, 2-359
?MBOOP value 2-370
?MBTC system call 2-358f
?MBWAIT value 2-370
?MCOBIT value 2-64f
?MCPID value 2-64
?MCRNG value 2-64
?MDAC offset 2-354f
?MDAL offset 2-354f
?MDCL offset 2-354, 2-353, 2-355
?MDDL offset 2-354, 2-353, 2-355
?MDDT offset 2-354, 2-353, 2-355
?MDID offset 2-354f
?MDIL offset 2-354, 2-353, 2-355
?MDLA offset 2-354, 2-353, 2-355
?MDLL offset 2-354f
?MDN0 offset 2-354, 2-353, 2-355
?MDN1 offset 2-354, 2-353, 2-355
?MDNL offset 2-354f
?MDNP offset 2-354f
?MDOP offset 2-354, 2-353, 2-355
?MDOX offset 2-354f
?MDP0 value 2-354
?MDPC offset 2-354, 2-353, 2-355
?MDPK offset 2-354, 2-353, 2-355
?MDPL offset 2-354, 2-353, 2-355
?MDPV value 2-352
?MDRE offset 2-354f
?MDRL offset 2-354, 2-353, 2-355
?MDRP offset 2-354, 2-353, 2-355
?MDRT offset 2-354, 2-353, 2-355
?MDUMP system call 2-360f
?MEM system call 2-362
?MEMI system call 2-363f
memory
  address 1-1
  management system calls (names of) 2-4
?MFSYM value 2-370
?MIFUN offset 2-369f
?MII1 value 2-370
?MII2 value 2-370
?MIID offset 2-369f
?MILD offset 2-369f
?MILN value 2-369
?MIOP offset 2-369f
?MIPHI offset 2-369f
?MIPLO offset 2-369f
?MIPUL offset 2-369f
?MIR1 offset 2-369f
?MIR2 offset 2-369f
?MIR3 offset 2-369f
?MIR4 offset 2-369f
?MIRES offset 2-369f
?MIRROR system call 2-365ff
?MIRROR_OVERRIDE value 2-367
?MIRROR_PKT.FUNC offset 2-366f
?MIRROR_PKT.LDID offset 2-366f
?MIRROR_PKT.LDNAME offset 2-366f
?MIRROR_PKT_LEN value 2-366
?MIRROR_PKT.OPTIONS offset 2-366f
?MIRROR_PKT.PKT_ID offset 2-366f
?MIRROR_PKT_PKTID value 2-367
?MIRROR_PKT.PUL offset 2-366f
?MIRROR_PKT.RES offset 2-366f
?MIRROR_PUL_PKTID value 2-368
?MIRROR_SYNCHRONIZE value 2-367
?MIRROR_SYNC_WAIT value 2-367
?MMAP value 2-353, 2-355
most significant bits 1-5
moving the pointer 2-577ff
?MPHIST system call 2-372ff
?MPHIST_HAPH offset 2-376
?MPHIST_HARAY offset 2-376
?MPHIST_HPRH offset 2-376
?MPHIST_HSBH offset 2-376
?MPHIST_HSIH offset 2-376
?MPHIST_HTTH offset 2-376
?MPHIST_PKT_PKTID value 2-375
?MPH_PID value 2-375
?MPH_PKT.FILLER offset 2-374f
?MPH_PKT.FUNC offset 2-374f
?MPH_PKT.HWEND offset 2-374f

?MPH_PKT.HWST offset 2-374f
?MPH_PKT.HWWDS offset 2-374f
?MPH_PKT.JPx offset 2-374, 2-376
?MPH_PKT.PCODE offset 2-374f
?MPH_PKT.PID offset 2-374f
?MPH_PKT.PKT_ID offset 2-374f
?MPH_PKT.PNAME offset 2-374f
?MPH_PKT.PRETURNED offset 2-374f
?MPH_PKT.PSUPPLIED offset 2-374f
?MPH_PNAME value 2-375
?MRDO value 2-353, 2-355
multiprocessor management system calls (names of) 2-12
multitasking system calls (names of) 2-10f
?MXFN value 2-830
?MXHN value 2-255, 2-847
?MXLPN value 2-71
?MXPL value 2-191f, 2-553f, 2-629, 2-683f
?MXUN value 2-252, 2-847
?MYTID system call 2-377
  example of, *see* program NEWTASK.SR

N

NAK character 2-664
NEWTASK.SR sample program B-30f, B-2
?NFKY offset 2-196
?NFLN value 2-196
?NFNM offset 2-196
?NFRS offset 2-196
?NFTP offset 2-196
normal return 1-2
?NPAL offset 2-206f
?NPAP offset 2-206f
?NPFW offset 2-206f
?NPKEY offset 2-206f
?NPLTH value 2-206
?NPNEN offset 2-206f
?NPNUM offset 2-206f
?NPPR offset 2-206f
?NPRS1 offset 2-206f
?NTRN value 2-713

O

?OANS value 2-385
?OBBQ value 2-401
?OBCD value 2-401
?OBCO value 2-401
?OBGM value 2-401
?OBHD value 2-401
?OBLD value 2-401
?OBLT value 2-401
?OBMI value 2-401
?OBPR value 2-401
?OBQU value 2-401
?OBUD value 2-401
?OBUT value 2-401
?OCIL offset 2-408f
?OCOL offset 2-408f
?OCR1 offset 2-408f
?OCR2 offset 2-408f
?OCR3 offset 2-408f
?OCR4 offset 2-408f
?OCR5 offset 2-408f
?OCR6 offset 2-408f
?OCR7 offset 2-408f
?OCR8 offset 2-408f
?OCR9 offset 2-408f
?OCRD offset 2-408f
?ODBS value 2-203
?ODBY offset 2-203
?ODBY value 2-202
?ODF1 offset 2-203
?ODF1 value 2-202
?ODHD offset 2-203
?ODHD value 2-202
?ODHS value 2-203
?ODIS system call 2-378
?ODMB value 2-203
?ODND value 2-203
?ODP0 value 2-203
?ODSEC offset 2-203
?ODSEC value 2-202
?ODST value 2-203
?ODTP value 2-203
?ODTRK offset 2-203
?ODTRK value 2-202
?OEBL system call 2-379
?OFCE value 2-384f, 2-387
?OFCR value 2-384f, 2-387
?OFE2 offset 2-404f
?OFEI offset 2-404f
?OFEO offset 2-404f
?OFER offset 2-404f
offset 1-3
?OFID value 2-405
?OFIN value 1-4, 2-384, 2-387
?OFIO value 2-384, 2-387
?OFOT value 2-384, 2-387
?OIBM value 2-385
?OIIL offset 2-411f
?OIN2 offset 2-411f
?OIND offset 2-411f
?OINL value 2-411
?OINP offset 2-411f
?OINR offset 2-411f
?OINT offset 2-411f
?OIOL offset 2-411f

Licensed Material - Property of Data General Corporation

?ONE2 offset 2-402f
?ONEI offset 2-402, 2-404
?ONEO offset 2-402, 2-404
?ONER offset 2-402f
?ONID value 2-403
?OOF2 offset 2-404f
?OOF3 offset 2-404f
?OOFE offset 2-404f
?OOFL value 2-404
?OOFN offset 2-404f
?OOFT offset 2-404f
?OOG2 offset 2-402f
?OOGT offset 2-402f
?OON2 offset 2-402f
?OON3 offset 2-402f
?OONE offset 2-402f
?OONL value 2-402
?OONN offset 2-402f
?OONT offset 2-402f
?OPAM value 2-42, 2-200
?OPCH offset 2-199
?OPD0 value 2-121
?OPD1 value 2-121
?OPD2 value 2-121
?OPDH value 2-42, 2-200
?OPDL value 2-42, 2-200
?OPDM value 2-42, 2-200
?OPEH offset 2-200
?OPEN 1-3
?OPEN system call 2-380ff
  example of, *see* program BOOMER.SR
  example of, *see* program DLIST.SR
  example of, *see* program FILCREATE.SR
  example of, *see* program HEAR.SR
  example of, *see* program INRING.SR
  example of, *see* program NEWTASK.SR
  example of, *see* program RINGLOAD.SR
  example of, *see* program WRITE.SR
opening a graphics window's pixel map 2-216ff
?OPER system call 2-399ff
operators
  ! 1-4
  B 1-4
  S 1-4
?OPEX commands
  access 2-418
  align 2-419
  allocate 2-420
  autokey 2-491
  batch_list 2-421
  batch_output 2-422
  binary 2-423
  brief 2-424
  cancel 2-425
  close 2-426
  consolestatus 2-427f
  continue 2-429
  cpl 2-430
  create 2-431
  defaultforms 2-432
  delete 2-433
  deregister 2-493
  disable 2-434
  dismounted 2-435
  elongate 2-436
  enable 2-437f
  even 2-439
  flush 2-440
  font 2-441
  forms 2-442
  halt 2-443
  headers 2-444
  hold 2-445
  limit 2-446
  logging 2-447f
  lpp 2-449
  mapper 2-450
  message 2-451
  modify 2-452
  mounted 2-453
  mountstatus 2-454ff
  open 2-457
  operator 2-458
  pause 2-459
  premount 2-460
  priority 2-461
  prompts 2-462
  purge 2-463
  qpriority 2-464f
  refused 2-466
  register 2-492
  registerstatus 2-494f
  release 2-467
  restart 2-468
  silence 2-469
  spoolstatus 2-470ff
  stack 2-473
  start 2-474f
  status 2-476ff
  stop 2-481
  terminate 2-482
  trailers 2-483
  unhold 2-484
  unitstatus 2-485f
  unlimit 2-487
  unsilence 2-488
  verbose 2-489
  xbias 2-490
?OPEX system call 2-413ff
?OPFC offset 2-200
?OPFL offset 2-200, 2-202
?OPIL offset 2-410f

?OPINFO code 2-400
?OPINFO function of ?OPER 2-411f
?OPK2 offset 2-400f
?OPKT offset 2-400f
?OPLT value 2-199f
?OPMD value 2-200
?OPME value 2-200
?OPNL value 2-400
?OPOFF code 2-400
?OPOFF function of ?OPER 2-404f
?OPON code 2-400
?OPON function of ?OPER 2-402f
?OPPH offset 2-199
?OPRCV code 2-400
?OPRCV function of ?OPER 2-408f
?OPRESP code 2-400
?OPRESP function of ?OPER 2-410f
?OPSEND code 2-400
?OPSEND function of ?OPER 2-406f
?OPSP offset 2-400f
?OPTY offset 2-199f
?OPXL value 2-202f
?OPXP bit 2-202
?OPXP value 2-200
?OPXS offset 2-203
?OPXS value 2-202
?ORC2 offset 2-408f
?ORC3 offset 2-408f
?ORC4 offset 2-408f
?ORCL value 2-408
?ORCN offset 2-408f
?ORCP offset 2-408f
?ORCQ offset 2-408f
?ORCS offset 2-408f
?ORCT offset 2-408f
?ORDS value 2-74f
?ORDY value 2-74f
?ORE2 offset 2-400f
?ORES offset 2-400f
?OREV offset 2-400f
?ORFX value 2-74f
?ORLC code 2-407
?ORLO code 2-407
?ORMNV code 2-407
?ORMU code 2-407
?ORP2 offset 2-410f
?ORP3 offset 2-410f
?ORP4 offset 2-410f
?ORPE offset 2-410f
?ORPL value 2-410
?ORPN offset 2-410f
?ORPP offset 2-410f
?ORPS offset 2-410f
?ORPT offset 2-410f
?ORSC offset 2-408f
?ORV value 2-74f
?OSID offset 2-406f
?OSIL offset 2-406f
?OSN2 offset 2-406f
?OSN3 offset 2-406f
?OSNF offset 2-406f
?OSNG code 2-407
?OSNL code 2-407
?OSNL value 2-406
?OSNN offset 2-406f
?OSNO code 2-407
?OSNP offset 2-406f
?OSNQ offset 2-406f
?OSNR offset 2-406f
?OSNT offset 2-406f
?OSOL offset 2-406f
?OSPI offset 2-400f
?OVEX system call 2-496
?OVKIL system call 2-497
?OVLOD system call 2-498ff
?OVREL system call 2-501

P

?PACDEV value 2-548f
packet address, *see* parameter packets
packets 1-3
packets, parameter 1-3
?PALW value 2-391
parameter packets 1-3
parametric coding 1-3
PARU.16.SR 1-3
PARU.32.SR 1-3, 1-9
PARU_LONG.SR 1-3, 1-9
PASCAL 1-11
?PBATCHP value 2-547, 2-549
?PBCHPRV value 2-547, 2-549
?PBLKS offset 2-558, 2-843
?PBLMEM value 2-548f
?PBLT value 2-587, 2-694
?PBRK offset 2-37
?PBRK value 2-531
?PBWSS value 2-548f
?PCAD offset 2-519f, 2-587f, 2-694
?PCAL offset 2-530, 2-533
?PCHPRI value 2-548f
?PCHTYP value 2-548f
?PCHUSER value 2-548f
?PCHWSSL value 2-549
?PCLASS system call 2-502ff
?PCL_PID value 2-504
?PCL_PKT.CLASS_ID offset 2-503f
?PCL_PKT.FUNC offset 2-503f
?PCL_PKT_LEN value 2-503
?PCL_PKT.PCODE offset 2-503f
?PCL_PKT.PID offset 2-503f

Licensed Material - Property of Data General Corporation

?PCL_PKT.PKT_ID offset 2-503f
?PCL_PKT_PKTID value 2-504
?PCL_PKT.PNAME offset 2-503f
?PCL_PKT.PROG_LOC offset 2-503f
?PCL_PKT.PSUPPLIED offset 2-503f
?PCL_PKT.USER_LOC offset 2-503f
?PCL_PNAME value 2-504
?PCNSPRV value 2-547, 2-549
?PCNX system call 2-505
?PCOMMNT value 2-548
?PCON offset 2-530, 2-533
?PCS1 offset 2-519ff
?PCS2 offset 2-519ff
?PCS3 offset 2-519ff
?PCS4 offset 2-519ff
?PCS5 offset 2-519ff
?PCS6 offset 2-519f
?PCS7 offset 2-519f
?PCS8 offset 2-519f
?PDEL mask 2-387
?PDEL value 2-384
?PDESLN offset 2-562, 2-848
?PDESLN value 2-558, 2-843
?PDFP offset 2-530, 2-534
?PDIR offset 2-530, 2-532
?PDISKLM value 2-547
?PDLOCY value 2-549
?PDMP value 2-532
?PFADW offset 2-513, 2-691
?PFAL offset 2-546, 2-550
?PFBI value 2-546
?PFBS value 2-531
?PFBY value 2-546
?PFCRE value 2-545
?PFDA value 2-531
?PFDB value 2-93, 2-531
?PFDEL value 2-545
?PFDL offset 2-546
?PFDl offset 2-550
?PFDP offset 2-546, 2-550
?PFER offset 2-546, 2-550
?PFEX value 2-531
?PFFC offset 2-544f
?PFFD offset 2-546
?PFFD 2-547ff
?PFFLG offset 2-513, 2-691f
?PFFLG value 2-690
?PFFO value 2-513, 2-690ff
?PFGNP value 2-545
?PFIAC value 2-545
?PFIH offset 2-513, 2-691
?PFLB value 2-544
?PFLE value 2-546
?PFLG offset 2-93, 2-530ff
?PFLNG value 2-513, 2-691
?PFNF offset 2-544f
?PFPID offset 2-513, 2-691
?PFPM value 2-531, 2-533
?PFPP value 2-531
?PFPR offset 2-544ff
?PFR1 offset 2-544f
?PFR3 offset 2-546, 2-550
?PFRDF value 2-545
?PFREN value 2-545
?PFRNG offset 2-513, 2-691
?PFRP value 2-83, 2-532
?PFRS value 2-83, 2-532
?PFRV offset 2-544f
?PFRW value 2-513, 2-691
?PFSE value 2-546
?PFTAC value 2-545
?PFUFD value 2-545
?PFUN offset 2-544f
?PFVER value 2-550
?PFWD value 2-546
?PFXP value 2-531
?PGBIN value 2-509
?PGLID value 2-509
?PGLIST system call 2-506
?PGLNM value 2-509
?PGNAME system call 2-507ff
PGN_ACT_LEN offset 2-508f
PGN_AVAIL_SIZE offset 2-508f
PGN_GROUP_NAME offset 2-508f
PGN_PGID offset 2-508f
?PGN_PKTID value 2-509
PGN_PKT_LEN value 2-508
PGN_REV_ID offset 2-508f
PGN_SUBPKT_IND offset 2-508f
?PGRID value 2-509
?PGRNM value 2-509
?PHRDPRV value 2-549
?PICCFN value 2-547
?PICROG value 2-547
?PIDS system call 2-510f
?PIFG offset 2-390f
?PIFP offset 2-530, 2-534
?PILN value 2-390
?PILRP offset 2-511
?PILTH value 2-511
?PIMXP offset 2-511
?PINTDIR value 2-549
?PIPC offset 2-530, 2-532
?PIPD offset 2-390f
?PIPI offset 2-390f
?PIPR offset 2-511
?PIRS offset 2-390f
?PIRV offset 2-390f
?PITI offset 2-390f
?PITOT offset 2-511
?PKR0 offset 2-545
?PKR0 value 2-119ff

?PKR1 value 2-683
PL/I 1-11
?PLFP offset 2-530, 2-534
?PLOGON value 2-547
?PLTH value 2-530, 2-539
?PMCTS value 2-549
?PMDIS offset 2-557, 2-561f
?PMDIS value 2-848
?PMEM offset 2-530, 2-532
?PMGSYS value 2-548f
?PMODPRV value 2-547, 2-549
?PMSDEN offset 2-557, 2-561
?PMTPF system call 2-512f
?PMXSONS value 2-547, 2-549
?PMYSONS value 2-548f
?PN1FLG value 2-550
?PN2FLG value 2-550
?PNAME system call 2-514f
?PNBLMEM value 2-548f
?PNBWSS value 2-548f
?PNCRYPT value 2-548
?PNM offset 2-530, 2-532
?PNVR value 2-391
?POFP offset 2-530, 2-534
pointer
  byte 1-5
  frame 1-2
?POLOCY value 2-550
position, bit 1-4
?PPASSWD value 2-547
?PPBLT value 2-519
?PPCR offset 2-530, 2-533
?PPDPMGR value 2-548f
?PPRCINF value 2-549
?PPRI offset 2-530, 2-532
?PPRNBLK value 2-548f
?PPRV offset 2-530, 2-533
?PPSUPP value 2-548f
?PPWDPRV value 2-547, 2-549
?PQLOCY value 2-550
?PRBB offset 2-519f, 2-522
?PRCL offset 2-249, 2-519f, 2-587f, 2-694
?PRCNX system call 2-516f
?PRCRYPT value 2-550
?PRDB system call 2-518ff
?PRES offset 2-587f, 2-694
?PRHRDPR value 2-550
?PRI system call 2-523
?PRIPR system call 2-524f
?PRKIL system call 2-526
?PRNH offset 2-17, 2-249, 2-519f, 2-586ff, 2-694
?PRNL offset 2-587f
?PROC extension packet 2-539f
?PROC functions
  creating offspring 2-537
  sending a CLI-like command line 2-536
  setting maximum CPU time 2-538
  setting the working set size 2-538
?PROC system call 2-527ff
  example of, *see* program CREATE_WINDOW.SR
  example of, *see* program SON.SR
process management system calls (names of) 2-5f
?PROFILE system call 2-543ff
program
  construction 1-9
  execution 1-9
programs, sample
  BOOMER.SR B-33ff, B-2
  CHECK.F77 B-45, B-2
  CREATE_WINDOW.SR B-49ff, B-2
  DIRCREATE.F77 B-42f, B-2
  DIRCREATE.SR 1-6ff
  DLIST.SR B-27ff, B-2
  FILCREATE.SR B-20f, B-2
  GATE.ARRAY.SR B-18, B-2
  HEAR.SR B-3ff, B-2
  INRING.SR B-17f, B-2
  NEWTASK.SR B-30f, B-2
  RINGLOAD.SR B-15f, B-2
  RUNTIME.SR B-12ff, B-2
  SON.SR B-9f, B-2
  SPEAK.SR B-7f, B-2
  TIMEOUT.SR B-38f, B-2
  WRITE.SR B-23ff, B-2
programming, assembly language 1-6ff
?PRPSSWD value 2-550
?PRRAPRV value 2-547, 2-549
?PRRDY system call 2-551
?PRSFTPR value 2-550
?PRSUS system call 2-552
?PSAL offset 2-559
?PSCH offset 2-557, 2-560
?PSCPL offset 2-557, 2-560
?PSCW offset 2-557, 2-560
?PSEN offset 2-557, 2-559
?PSEX offset 2-557, 2-560f
?PSF2 offset 2-557, 2-559
?PSF3 offset 2-557, 2-560
?PSF4 offset 2-557, 2-560
?PSF5 offset 2-557, 2-560
?PSFA offset 2-557, 2-561
?PSFL offset 2-557, 2-559
?PSFP offset 2-557, 2-559
?PSFTPRV value 2-549
?PSHSH offset 2-558, 2-843
?PSHST offset 2-558, 2-843
?PSHSZ offset 2-558, 2-843
?PSIH offset 2-557, 2-561
?PSLFA offset 2-557, 2-561

Licensed Material - Property of Data General Corporation

?PSLIST system call 2-553f
?PSLTH value 2-557
?PSMX offset 2-557, 2-561
?PSNM offset 2-530, 2-532
?PSNR offset 2-557, 2-559
?PSNS offset 2-557, 2-559
?PSOPIO value 2-547, 2-549
?PSPD offset 2-557, 2-560
?PSPH offset 2-557, 2-560
?PSPP value 2-559, 2-844
?PSPR offset 2-557, 2-560
?PSPRST offset 2-558, 2-843
?PSPV offset 2-557, 2-560
?PSQF offset 2-557, 2-559
?PSRH offset 2-557, 2-560
?PSSL offset 2-557, 2-561
?PSSN offset 2-557, 2-559
?PSSP value 2-559, 2-844
?PSST offset 2-557, 2-559
?PSTA value 2-559, 2-844
?PSTAT system call 2-555ff
?PSTB value 2-559, 2-844
?PSTC value 2-559, 2-844
?PSTI offset 2-17, 2-519f, 2-522, 2-587f, 2-693f
?PSTO offset 2-519f, 2-587f, 2-694
?PSUSER value 2-548f
?PSWM offset 2-557, 2-561
?PSWS offset 2-557, 2-561
?PTRDEVICE system call 2-564ff
?PTRDEV_EVENTS_ACTIVATION value 2-572
?PTRDEV_EVENTS_BTN_DOWN value 2-572, 2-579
?PTRDEV_EVENTS_BTN_UP value 2-572, 2-579
?PTRDEV_EVENTS_DBL_CLICK value 2-572
?PTRDEV_EVENTS_DEACTIVATION value 2-572
?PTRDEV_EVENTS_ENTER_WINDOW value 2-572
?PTRDEV_EVENTS_EXIT_WINDOW value 2-572
?PTRDEV_EVENTS_MOVEMENT value 2-572, 2-579
?PTRDEV_GENERATE_EVENT function code 2-565
?PTRDEV_GENERATE_EVENT function 2-577ff
?PTRDEV_GENERATE_EVENT_PKT_ID value 2-579
?PTRDEV_GEN_EVENT.BUTTON offset 2-578f
?PTRDEV_GEN_EVENT.EVENT offset 2-578f
?PTRDEV_GEN_EVENT.FLAGS offset 2-578f
?PTRDEV_GEN_EVENT.FLAGS.X value 2-579
?PTRDEV_GEN_EVENT.FLAGS.Y value 2-579
?PTRDEV_GEN_EVENT_LEN value 2-578
?PTRDEV_GEN_EVENT.PKT_ID offset 2-578f
?PTRDEV_GEN_EVENT.POS.X offset 2-578f
?PTRDEV_GEN_EVENT.POS.Y offset 2-578f
?PTRDEV_GEN_EVENT.VIRTUAL value 2-579
?PTRDEV_GET_PTR_STATUS function code 2-575f, 2-565
?PTRDEV_GSTATUS.BTNS offset 2-575f
?PTRDEV_GSTATUS.DELTA.X offset 2-575f
?PTRDEV_GSTATUS.DELTA.Y offset 2-575f
?PTRDEV_GSTATUS.EVTS offset 2-575f
?PTRDEV_GSTATUS_LEN value 2-575
?PTRDEV_GSTATUS.PKT_ID offset 2-575f
?PTRDEV_GSTATUS_PKTID value 2-576
?PTRDEV_GSTATUS.PTR_SHAPE offset 2-575f
?PTRDEV_GSTATUS.PTR_STATE offset 2-575f
?PTRDEV_LAST_EVENT function code 2-571ff, 2-565
?PTRDEV_LEVENT.BTN offset 2-572
?PTRDEV_LEVENT.EVTS offset 2-572
?PTRDEV_LEVENT_LEN value 2-572
?PTRDEV_LEVENT.PKT_ID offset 2-572
?PTRDEV_LEVENT_PKTID value 2-572
?PTRDEV_LEVENT.PTR_LOC.X offset 2-572
?PTRDEV_LEVENT.PTR_LOC.Y offset 2-572
?PTRDEV_PKT.CHAN_NUM offset 2-566f
?PTRDEV_PKT.FLAGS offset 2-566f, 2-566
?PTRDEV_PKT.FLAGS.IN_CHAN value 2-567
?PTRDEV_PKT.FLAGS.IN_PATH value 2-567
?PTRDEV_PKT.FLAGS.IN_WIND_ID value 2-567
?PTRDEV_PKT.FUNC offset 2-566f, 2-565
?PTRDEV_PKT_LEN value 2-566
?PTRDEV_PKT.PATH offset 2-566f
?PTRDEV_PKT.PATH_BUF_LEN offset 2-566, 2-568
?PTRDEV_PKT.PATH_LEN offset 2-566, 2-568
?PTRDEV_PKT.PKT_ID offset 2-566f
?PTRDEV_PKT_PKTID value 2-567
?PTRDEV_PKT.SUBPKT offset 2-566, 2-568
?PTRDEV_PKT.WIND_ID offset 2-566, 2-568
?PTRDEV_PTR_SHAPE_ARROW value 2-574, 2-576
?PTRDEV_PTR_SHAPE_DEFAULT value 2-574
?PTRDEV_PTR_SHAPE_FINGER value 2-574, 2-576
?PTRDEV_PTR_SHAPE_FXHAIR value 2-574, 2-576
?PTRDEV_PTR_SHAPE_HOURGLASS value 2-574, 2-576
?PTRDEV_PTR_SHAPE_SXHAIR value 2-574, 2-576
?PTRDEV_PTR_STATE_DISABLE value 2-574, 2-576
?PTRDEV_PTR_STATE_ENABLE value 2-574, 2-576
?PTRDEV_SET_DELTA function code 2-570f, 2-565
?PTRDEV_SET_DELTA.DELTA.X value 2-570
?PTRDEV_SET_DELTA.DELTA.Y value 2-570
?PTRDEV_SET_DELTA.FLAGS value 2-570
?PTRDEV_SET_DELTA.FLAGS.X value 2-571

?PTRDEV_SET_DELTA.FLAGS.Y value 2-571
?PTRDEV_SET_DELTA_LEN value 2-570
?PTRDEV_SET_DELTA.PKT_ID offset 2-570
?PTRDEV_SET_DELTA_PKTID value 2-571
?PTRDEV_SET_EVENTS function code 2-565, 2-568
?PTRDEV_SET_EVTS.BTNS offset 2-568, 2-570
?PTRDEV_SET_EVTS.BTNS.ONE value 2-570
?PTRDEV_SET_EVTS.BTNS.THREE value 2-570
?PTRDEV_SET_EVTS.BTNS.TWO value 2-570
?PTRDEV_SET_EVTS.EVTS offset 2-568f
?PTRDEV_SET_EVTS.EVTS.BTN_DOWN value 2-569
?PTRDEV_SET_EVTS.EVTS.BTN_UP value 2-569
?PTRDEV_SET_EVTS.EVTS.DBL_CLICK value 2-569
?PTRDEV_SET_EVTS.EVTS.MOVEMENT value 2-569
?PTRDEV_SET_EVTS.EVTS.WINDOW value 2-569
?PTRDEV_SET_EVTS_LEN value 2-568
?PTRDEV_SET_EVTS.PKT_ID offset 2-568
?PTRDEV_SET_EVTS_PKTID value 2-568
?PTRDEV_SET_POINTER function code 2-573f, 2-565
?PTRDEV_SET_PTR.FLAGS offset 2-573f
?PTRDEV_SET_PTR.FLAGS.SHAPE value 2-574
?PTRDEV_SET_PTR.FLAGS.STATE value 2-574
?PTRDEV_SET_PTR_LEN value 2-573
?PTRDEV_SET_PTR.PKT_ID offset 2-573f
?PTRDEV_SET_PTR_PKTID value 2-574
?PTRDEV_SET_PTR.SHAPE offset 2-573f
?PTRDEV_SET_PTR.STATE offset 2-573f
?PTWO value 2-391
?PUDAH offset 2-594
?PUDAL offset 2-594
?PUDCN offset 2-594
?PUDFW offset 2-594
?PUDLT value 2-594
?PUIPCS value 2-548f
?PUL_MAX_NAMES value 2-368, 2-831
?PUL_PKT.COUNT offset 2-368, 2-831
?PUL_PKT_LEN value 2-368, 2-831
?PUL_PKT.PKT_ID offset 2-368, 2-831
?PUL_PKT.ULN1 offset 2-368, 2-831
?PUL_PKT.ULN2 offset 2-368, 2-831
?PUL_PKT.ULN3 offset 2-368, 2-831
?PUL_PKT.ULN4 offset 2-368, 2-831
?PUL_PKT.ULN5 offset 2-368, 2-831
?PUL_PKT.ULN6 offset 2-368, 2-831
?PUL_PKT.ULN7 offset 2-368, 2-831
?PUL_PKT.ULN8 offset 2-368, 2-831
?PUNM offset 2-530, 2-533
?PUSPR offset 2-558, 2-843
?PVCNPRV value 2-547, 2-549
?PVDV value 2-535
?PVEX value 2-535
?PVIP value 2-535
?PVPC value 2-535
?PVPP value 2-535
?PVPR value 2-535
?PVSP value 2-535
?PVSU value 2-535
?PVTY value 2-535
?PVUI value 2-535
?PVWM value 2-535
?PVWS value 2-535
?PWBP offset 2-581f
?PWDCRYP system call 2-580ff
?PWFW offset 2-581f
?PWLO offset 2-581
?PWLO value 2-582
?PWMI offset 2-530, 2-534
?PWOW value 2-582
?PWRB system call 2-518ff, 2-583
?PWRV offset 2-581f
?PWSON value 2-549
?PWSS offset 2-530, 2-533
?PWSZ value 2-581
?PXCPU offset 2-539f
?PXFLG offset 2-539f
?PXLE value 2-539
?PXLLOC offset 2-539f
?PXPGN offset 2-539f
?PXPUN offset 2-539f
?PXSPI offset 2-539f
?PXULOC offset 2-539f
?PXUPID offset 2-539f

R

?RCALL system call 2-584
?RCHAIN system call 2-585
?RCID value 2-409
?RDAC value 2-773
?RDB system call 2-586ff
?RDUDA system call 2-593f
?READ system call 2-595ff
  example of, *see* program BOOMER.SR
  example of, *see* program FILCREATE.SR
  example of, *see* program RUNTIME.SR
  example of, *see* program WRITE.SR
reading from a palette 2-230f
reading from a window's palette 2-818f
?REC system call 2-612
  example of, *see* program BOOMER.SR
?RECNW system call 2-613
?RECREATE system call 2-614
?REGISTER system call 2-615ff
?REGISTER_GLOBAL_NAME offset 2-616f

Licensed Material - Property of Data General Corporation

?REGISTER_PKTID value 2-617
?REGISTER_PKT_LEN value 2-616
?REGISTER_RESV offset 2-616f
?REGISTER_REV_ID offset 2-616f
?REGISTER_SUBPKT_IND offset 2-616f
?REGISTER_TARGET_NAME offset 2-616f
?RELEASE system call 2-618
?RENAME system call 2-619f
reporting, error 1-10
?RESCHED system call 2-621
reserved symbols 1-6
?RESIGN system call 2-622
resuming output to a suspended window 2-801
return
  error 1-2
  normal 1-2
?RETURN system call 2-623f, 1-2, 1-7f, 1-10
  example of, *see* program BOOMER.SR
  example of, *see* program CHECK.F77
  example of, *see* program DLIST.SR
  example of, *see* program FILCREATE.SR
  example of, *see* program HEAR.SR
  example of, *see* program NEWTASK.SR
  example of, *see* program RINGLOAD.SR
  example of, *see* program RUNTIME.SR
  example of, *see* program SON.SR
  example of, *see* program SPEAK.SR
  example of, *see* program TIMEOUT.SR
  example of, *see* program WRITE.SR
?RFAB value 2-285, 2-287, 2-623f
?RFCF 1-8
?RFCF value 2-285, 2-287, 2-623f
?RFEC 1-8
?RFEC value 2-285, 2-287, 2-623
?RFER 1-8
?RFER value 2-285, 2-287, 2-623f
?RFWA value 2-285, 2-287, 2-623f
?RINGLD system call 2-625f
  example of, *see* program RINGLOAD.SR
RINGLOAD.SR sample program B-15f, B-2
?RNAME system call 2-627
?RNGBP offset 2-629
?RNGLB offset 2-629
?RNGNM offset 2-629
?RNGPL value 2-629
?RNGPR system call 2-628f
?RNGST system call 2-630
?RPAGE system call 2-631f
?RSID value 2-411
?RTDS value 2-384, 2-599
?RTDY value 2-384, 2-599
?RTFX value 2-384, 2-599
?RTUN value 2-384, 2-599
?RTVB value 2-384, 2-599
?RTVR value 2-384, 2-599
RUNTIME.SR sample program B-12ff, B-2
?RUNTM system call 2-633ff
  example of, *see* program RUNTIME.SR
?RVBPL value 2-775
?RVBPX value 2-775
RVI character 2-664
?RVWPL value 2-775

S

S operator 1-4
?SACK value 2-700
?SACL system call 2-636ff
?SACP offset 2-162f, 2-164f
?SAFM value 2-588
?SAK0 value 2-700
?SAK1 value 2-700
sample programs
  BOOMER.SR B-33ff, B-2
  CHECK.F77 B-45, B-2
  CREATE_WINDOW.SR B-49ff, B-2
  DIRCREATE.F77 B-42f, B-2
  DIRCREATE.SR 1-6ff
  DLIST.SR B-27ff, B-2
  FILCREATE.SR B-20f, B-2
  GATE.ARRAY.SR B-18, B-2
  HEAR.SR B-3ff, B-2
  INRING.SR B-17f, B-2
  NEWTASK.SR B-30f, B-2
  RINGLOAD.SR B-15f, B-2
  RUNTIME.SR B-12ff, B-2
  SON.SR B-9f, B-2
  SPEAK.SR B-7f, B-2
  TIMEOUT.SR B-38f, B-2
  WRITE.SR B-23ff, B-2
?SASC value 2-660, 2-665
?SATR system call 2-639f
?SBER offset 2-675f
?SBIAS system call 2-641f
?SBSC value 2-661
?SBUL offset 2-712
?SBUP offset 2-699, 2-701, 2-712, 2-714
?SBYC offset 2-699, 2-701, 2-712, 2-714
?SBYM offset 2-699, 2-702, 2-712, 2-714
?SCHN offset 2-659f
?SCHR system call 2-643f
?SCIT value 2-660
?SCLOSE system call 2-647f
?SCON value 2-713
SCP note 2-787
?SCPS offset 2-165
?SCRC value 2-660
?SCSH offset 2-164f
?SCSL offset 2-164f
?SDAC value 2-701, 2-706, 2-713
?SDAD offset 2-699, 2-702

?SDAY system call 2-649
?SDBL system call 2-650
?SDCN value 2-651
?SDCU offset 2-162
?SDDN value 2-651
?SDEH offset 2-165
?SDEL offset 2-165
?SDET value 2-714
?SDIS value 2-714
?SDLM system call 2-651f
?SDMD value 2-660
?SDPOL system call 2-653ff
?SDPP value 2-660
?SDPR value 2-660
?SDRT system call 2-656f
?SDSC value 2-660
?SDTI value 2-651f
?SDTO value 2-651f
?SDTP value 2-651f
?SEBC value 2-660
?SEBL system call 2-658ff
?SECHR system call 2-667f
?SEFH offset 2-164f
?SEFL offset 2-164f
?SEFM offset 2-164f
?SEFW offset 2-164f
?SEID value 2-407
selecting the pointer events you want 2-568ff
?SELN value 2-659
?SEND system call 2-669f
  example of, *see* program DLIST.SR
sending a window or group to the back 2-799
?SEOT value 2-714
?SEPR value 2-660, 2-665
?SERMSG system call 2-671f
?SERT system call 2-656f, 2-673
?SERVE system call 2-674
setting a bit 1-4
setting a window's permanence status 2-802
setting a window's title 2-806
setting the clip rectangle 2-223ff
setting the windowing user interface 2-802ff
?SFAH offset 2-164f
?SFAL offset 2-164f
?SGES system call 2-675f
?SGLN value 2-675
?SHFS offset 2-164
?SHOP value 2-384
?SHSP value 2-661
?SIDX offset 2-164f
?SIEX value 2-682
significant bits
  least 1-5
  most 1-5
?SIGNL system call 2-677f
?SIGWT system call 2-679f
?SIID offset 2-682
?SILN offset 2-682
?SIMM offset 2-682
?SINFO system call 2-681ff
?SINT value 2-714
?SIOS offset 2-682
?SIRL offset 2-712, 2-714
?SIRN offset 2-682
?SIRS offset 2-682
?SITB value 2-700, 2-706, 2-713
?SLAU offset 2-164f
?SLIST system call 2-684
?SLRC value 2-660, 2-665
?SLTH value 2-162f, 2-164f
?SMCH offset 2-530, 2-534
?SMDI offset 2-659, 2-661
?SMIL offset 2-164f
?SMSH offset 2-164
?SMSL offset 2-164
?SNAK value 2-700
?SNKC offset 2-675f
?SNPR value 2-660, 2-665
?SOAL offset 2-686f
?SOFP offset 2-686f
?SOFW offset 2-686f
SOH character 2-664
?SOHB value 2-700, 2-706, 2-713
?SOKEY offset 2-686f
?SOLTH value 2-686
SON.SR sample program B-9f, B-2
?SONEN offset 2-686f
?SONS system call 2-685ff
?SOPEN system call 2-688f
?SOPN offset 2-164f
?SOPPF system call 2-690ff
?SOPR value 2-660, 2-665
?SORP offset 2-686f
?SOSON offset 2-686f
?SOSP offset 2-686f
?SPAGE system call 2-693ff
?SPAR value 2-609
SPEAK.SR sample program B-7f, B-2
specifying a pointer delta 2-570f
?SPET value 2-701
?SPLR value 2-700, 2-706
?SPNH offset 2-163
?SPNK value 2-701
?SPNL offset 2-163
?SPOS system call 2-696f
  example of, *see* program WRITE.SR
?SPRO value 2-693f
?SPRV value 2-701
?SPTM offset 2-81
?SRCV system call 2-698ff
?SRES offset 2-699, 2-701, 2-712, 2-714
?SRVI value 2-700

Licensed Material - Property of Data General Corporation

?SSHPT system call 2-708
?SSID system call 2-709
?SSIS offset 2-699ff, 2-712ff
?SSLR value 2-700, 2-706
?SSND system call 2-710ff
?SSNL value 2-699, 2-712
?SSTI offset 2-659ff
?SSTO offset 2-676
?SSTS offset 2-162f, 2-164ff
?STAH offset 2-164f
?STAL offset 2-164f
standard format for system calls 1-2
?STCH offset 2-162f, 2-164f
?STCL offset 2-162f, 2-164f
?STIM offset 2-162f, 2-164f
?STMAP system call 2-720ff
?STMH offset 2-164f
?STML offset 2-164f
?STOC offset 2-659, 2-661, 2-666
?STOD system call 2-723
?STOM system call 2-724f
?STOV offset 2-699, 2-702, 2-712, 2-714
?STTD value 2-714
?STTO offset 2-675
STX character 2-664
?STXB value 2-700, 2-706, 2-713
?STYP offset 2-162f, 2-164f
?SUPROC system call 2-726f
?SUS system call 2-728
?SUSER system call 2-729f
suspending window output 2-801
?SWAK value 2-701, 2-714
?SYLOG system call 2-731f
symbols, reserved 1-6
SYN character 2-665
SYSID.32.SR 1-9
:SYSLOG file 2-731f
:SYSLOG 2-338
?SYSPRV system call 2-733ff
?SYSPRV.FLAGS.CALLER value 2-735
?SYSPRV.FLAGS.CALLER_EXCL value 2-735
?SYSPRV.FLAGS.OTHERS value 2-735
?SYSPRV.FLAGS.OTHERS_EXCL value 2-735
?SYSPRV_ENTER value 2-735
?SYSPRV_ENTER_EXCL value 2-735
?SYSPRV_GET value 2-735
?SYSPRV_LEAVE value 2-735
?SYSPRV_PKT.FLAGS offset 2-734f
?SYSPRV_PKT.FUNC offset 2-734f
?SYSPRV_PKT_LEN value 2-734f
?SYSPRV_PKT.MODE offset 2-734f
?SYSPRV_PKT.PKT_ID offset 2-734f
?SYSPRV_PKT_PKTID value 2-735
?SYSPRV_PKT.SUBPKT offset 2-734f
?SYSPRV_SPROCESS value 2-735
?SYSPRV_SUSER value 2-735
?SYSPRV_SYSMGR value 2-735
system calls 1-1
  16-bit process (names of) 2-15
  AOS/DVS (names of) 2-4
  AOS/VS system resources (names of) 2-13
  bisynchronous communications (names of) 2-14
  class scheduling (names of) 2-13
  connection management (names of) 2-12
  debugging (names of) 2-8f
  file creation and management (names of) 2-7
  file input/output (names of) 2-8f
  interprocess communications (names of) 2-11
  memory management (names of) 2-4
  multiprocessor management (names of) 2-12
  multitasking (names of) 2-10f
  process management (names of) 2-5f
  standard format for 1-2
  user devices (names of) 2-14
  windowing (names of) 2-10
system log file 2-338f

T

?T32T code 2-286, 2-289
table, delimiter 2-389
?TABR code 2-286
?TALOCK value 2-764
?TAOS code 2-286f
task identifier, unique 2-677, 2-679, 2-768f
?TASK system call 2-736ff
  example of, *see* program BOOMER.SR
  example of, *see* program NEWTASK.SR
?TATH offset 2-71
?TBCX code 2-286f
?TBCX value 2-81
?TBLT value 2-71
?TCCX code 2-286f
?TCIN code 2-286f
?TCTH offset 2-71
?TDFL value 2-751, 2-755
?TEFH offset 2-249
?TEFM offset 2-249
?TEFW offset 2-249
?TERM system call 2-745f
  example of, *see* program SPEAK.SR
?TEXT code 2-285f, 2-289
?TIDSTAT system call 2-747
time value format 2-54
TIMEOUT.SR sample program B-38f, B-2
?TLOCK system call 2-748f
?TLTH value 2-249
?TM0H offset 2-754, 2-756
?TM0L offset 2-754, 2-756
?TM1H offset 2-754, 2-756
?TM1L offset 2-754, 2-756

?TM2H offset 2-754, 2-756
?TM2L offset 2-754, 2-756
?TM3H offset 2-754, 2-757
?TM3L offset 2-754, 2-757
?TM6 value 2-752, 2-756
?TMAC0 offset 2-751f
?TMAC1 offset 2-751f
?TMAC2 offset 2-751f
?TMAC3 offset 2-751f
?TMBH offset 2-755, 2-757
?TMBL offset 2-755, 2-757
?TMBLK offset 2-751, 2-753
?TMCH offset 2-754, 2-757
?TMCL offset 2-754, 2-757
?TMCPC offset 2-751, 2-753
?TMCPU offset 2-751, 2-753
?TMDH offset 2-755, 2-757
?TMDL offset 2-755, 2-757
?TMFH offset 2-755, 2-758
?TMFL offset 2-755, 2-758
?TMGH offset 2-755, 2-757
?TMGL offset 2-755, 2-757
?TMLH offset 2-754, 2-757
?TMLL offset 2-754, 2-757
?TMMLG offset 2-751, 2-753, 2-755, 2-758
?TMMSG offset 2-751, 2-753, 2-755, 2-758
?TMPGD offset 2-751, 2-753
?TMPGF offset 2-751, 2-753
?TMPGS offset 2-751, 2-753
?TMPID offset 2-751f, 2-754, 2-756
?TMPRV offset 2-751f, 2-754, 2-756
?TMR2 offset 2-751f, 2-754, 2-756
?TMR3 offset 2-751f, 2-754, 2-756
?TMR4 offset 2-751, 2-753, 2-755, 2-758
?TMR5 offset 2-751, 2-753, 2-755, 2-758
?TMR6 offset 2-751, 2-753, 2-755, 2-758
?TMR7 offset 2-751, 2-753, 2-755, 2-758
?TMR8 offset 2-751, 2-753, 2-755, 2-758
?TMR9 offset 2-751, 2-753, 2-755, 2-758
?TMRS offset 2-751f, 2-754, 2-756
?TMSEC offset 2-751, 2-753
?TMSG system call 2-750ff
?TMSH offset 2-754, 2-757
?TMSL offset 2-754, 2-757
?TMTCD offset 2-751, 2-753, 2-755, 2-758
?TMTH offset 2-71
?TMUPD offset 2-751f, 2-754, 2-756
?TMXTC offset 2-751f, 2-754, 2-756
?TMYRING value 2-764
?TPID system call 2-759
?TPLN value 2-289
?TPORT system call 2-760
?TR32 code 2-286
?TRAN value 2-700, 2-706, 2-713
transferring data between pixel maps and disk files 2-227ff
?TRAP code 2-286
?TRCON system call 2-761f
?TRPE value 2-189, 2-668
?TRUNCATE system call 2-763, B-40
?TSELF code 2-286f
?TSUP code 2-286f
TTD character 2-665
TTI note 2-595
TTO note 2-595
?TTY value 2-176, 2-178, 2-646
?TUNLOCK system call 2-764f

U

?UBLPR system call 2-766f
?UDBM value 2-261f
?UDDA value 2-262
?UDDB value 2-261f
?UDDC value 2-261f
?UDDD value 2-262
?UDDE value 2-262
?UDDF value 2-262
?UDDG value 2-262
?UDDH value 2-262
?UDDI value 2-262
?UDDJ value 2-262
?UDDK value 2-262
?UDDL value 2-262
?UDDM value 2-262
?UDDN value 2-262
?UDDO value 2-262
?UDDP value 2-262
?UDDRS offset 2-259
?UDELTH value 2-260
?UDID offset 2-261f
?UDLN value 2-259f
?UDLTH value 2-260
?UDNS value 2-262
?UDRS offset 2-259
?UDVBX offset 2-259f
?UDVIS offset 2-259f
?UDVMS offset 2-259f
?UDVXM offset 2-259f
?UIDSTAT system call 2-768f
?ULI1 value 2-836
?ULI2 value 2-371
?ULI3 value 2-371, 2-836
?ULN1 offset 2-371, 2-836
?ULN2 offset 2-371, 2-836
?ULN3 offset 2-371, 2-836
?ULN4 offset 2-371, 2-836
?ULN5 offset 2-371, 2-836
?ULN6 offset 2-371, 2-836
?ULN7 offset 2-371, 2-836
?ULN8 offset 2-371, 2-836

Licensed Material - Property of Data General Corporation

?ULNC offset 2-371, 2-836
?ULPHI offset 2-371, 2-836
?ULPLO offset 2-371, 2-836
?ULR0 offset 2-371, 2-836
?ULR1 offset 2-371, 2-836
?ULR2 offset 2-371, 2-836
?ULR3 offset 2-371, 2-836
?ULR4 offset 2-371, 2-836
?ULR5 offset 2-371, 2-836
?ULR6 offset 2-371, 2-836
?ULR7 offset 2-371, 2-836
?ULR8 offset 2-371, 2-836
unique task identifier 2-677, 2-679, 2-768f
unmapping a pixel map from your program's address space 2-229
?UNMR value 2-353, 2-355
?UNWIND system call 2-770
?UNWIRE system call 2-771
?UPDATE system call 2-772
user devices system calls (names of) 2-14
?UTID offset 2-769
?UTLEN value 2-769
?UTPRI offset 2-769
?UTSTAT offset 2-769
?UUID offset 2-769

V

?VALAD system call 2-773
?VALIDATE system call 2-774ff
?VBITM value 2-355
?VCUST system call 2-778
?VDELIM offset 2-775
?VERROR offset 2-775
?VFUNC offset 2-775
?VLAC value 2-773
?VLENGTH offset 2-775
?VPOINTER offset 2-775
?VRCUST system call 2-779
?VRES offset 2-775
?VRESD offset 2-775
?VRING offset 2-775
?VRNBR value 2-775

W

WACK character 2-665
?WALKBACK system call 2-780f

?WDELAY system call 2-782
example of, *see* program HEAR.SR
example of, *see* program SPEAK.SR
example of, *see* program TIMEOUT.SR
?WHIST system call 2-783ff
?WINDOW system call 2-786ff
example of, *see* program CREATE_WINDOW.SR
windowing system calls (names of) 2-10
?WIN_BORDER_TYPE_DEFAULT value 2-796, 2-804f
?WIN_BORDER_TYPE_INTELLIGENT value 2-796, 2-804f
?WIN_BORDER_TYPE_LINE value 2-796, 2-804f
?WIN_BORDER_TYPE_NONE value 2-796, 2-804f
?WIN_BORDER_TYPE_TITLE value 2-796, 2-804f
?WIN_CRE.BORDER_TYPE offset 2-792, 2-796
?WIN_CRE.FLAGS offset 2-792f, 2-794
?WIN_CRE.FLAGS.CREATE_SON value 2-793
?WIN_CRE.FLAGS.INPUT_BUF_SZ value 2-794
?WIN_CRE.FLAGS.NEW_GRP value 2-793
?WIN_CRE.FLAGS.PIXEL_DEPTH value 2-793
?WIN_CRE.FLAGS.UNHIDE_WINDOW value 2-793
?WIN_CRE.FLAGS.VIEW_ORG.X value 2-793
?WIN_CRE.FLAGS.VIEW_ORG.Y value 2-793
?WIN_CRE.FLAGS.VT_SZ_X value 2-793
?WIN_CRE.FLAGS.VT_SZ_Y value 2-793
?WIN_CRE.INPUT_BUF_SZ offset 2-792, 2-795
?WIN_CRE.PAL_TYPE offset 2-792, 2-795
?WIN_CRE.PIXEL_DEPTH offset 2-792, 2-795
?WIN_CRE.PKT_ID offset 2-792f
?WIN_CRE.RSV offset 2-792
?WIN_CRE.TITLE offset 2-792, 2-795
?WIN_CRE.TITLE_BUF_LEN offset 2-792, 2-795
?WIN_CRE.VIEW_ORG.X offset 2-792, 2-796
?WIN_CRE.VIEW_ORG.Y offset 2-792, 2-796
?WIN_CRE.VT_SZ.X offset 2-792, 2-794
?WIN_CRE.VT_SZ.Y offset 2-792, 2-794
?WIN_CRE.VT_TYPE offset 2-792, 2-794
?WIN_CRE.WNAME offset 2-792, 2-795
?WIN_CRE.WNAME_BUF_LEN offset 2-792, 2-795
?WIN_CREATE_WINDOW function code 2-788
?WIN_CRE_LEN_REV1 value 2-792
?WIN_CRE_PKTID_REV1 value 2-793
?WIN_DEFINE_PORTS function code 2-797ff, 2-788
?WIN_DELETE_WINDOW function code 2-796, 2-788
?WIN_DEVICE_STATUS function code 2-813ff, 2-789
?WIN_DEVSTAT.LANG_ID offset 2-813f
?WIN_DEVSTAT_LEN_REV1 value 2-813
?WIN_DEVSTAT.PIXEL_DEPTH offset 2-813f
?WIN_DEVSTAT.PKT_ID offset 2-813f
?WIN_DEVSTAT_PKTID_REV1 value 2-814
?WIN_DEVSTAT.PTR_TYPE offset 2-813f
?WIN_DEVSTAT.RSV offset 2-813f

?WIN_DEVSTAT.SZ.X offset 2-813f
?WIN_DEVSTAT.SZ.Y offset 2-813f
?WIN_DISABLE_KEYBOARD function code 2-801, 2-789
?WIN_DPORT.FLAGS offset 2-797f
?WIN_DPORT.FLAGS.SCAN_ORG_X value 2-798
?WIN_DPORT.FLAGS.SCAN_ORG_Y value 2-798
?WIN_DPORT.FLAGS.SCAN_SZ_X value 2-798
?WIN_DPORT.FLAGS.SCAN_SZ_Y value 2-798
?WIN_DPORT.FLAGS.VIEW_ORG_X value 2-798
?WIN_DPORT.FLAGS.VIEW_ORG_Y value 2-798
?WIN_DPORT_LEN value 2-797
?WIN_DPORT.PKT_ID offset 2-797f
?WIN_DPORT_PKTID value 2-798
?WIN_DPORT.SCAN_ORG.X offset 2-797f
?WIN_DPORT.SCAN_ORG.Y offset 2-797f
?WIN_DPORT.SCAN_SZ.X offset 2-797f
?WIN_DPORT.SCAN_SZ.Y offset 2-797f
?WIN_DPORT.VIEW_ORG.X offset 2-797f
?WIN_DPORT.VIEW_ORG.Y offset 2-797f
?WIN_ENABLE_KEYBOARD function code 2-789
?WIN_GET_TITLE function code 2-806f, 2-789
?WIN_GET_USER_INTERFACE function code 2-804ff, 2-789
?WIN_GET_WINDOW_ID function code 2-813, 2-789
?WIN_GET_WINDOW_NAME function code 2-807, 2-789
?WIN_GINT.BORDER_TYPE offset 2-804f
?WIN_GINT.FLAGS offset 2-804
?WIN_GINT.FLAGS.ELEVATORS value 2-805
?WIN_GINT.FLAGS.FULL_REV value 2-805
?WIN_GINT.FLAGS.HELP value 2-805
?WIN_GINT.FLAGS.MOVE_X value 2-805
?WIN_GINT.FLAGS.MOVE_Y value 2-805
?WIN_GINT.FLAGS.SCROLL_X value 2-805
?WIN_GINT.FLAGS.SCROLL_Y value 2-805
?WIN_GINT.FLAGS.SZ_X value 2-805
?WIN_GINT.FLAGS.SZ_Y value 2-805
?WIN_GINT.FLAGS.WIRE_BRDRS value 2-805
?WIN_GINT_LEN value 2-804
?WIN_GINT_PKT_ID value 2-804
?WIN_GINT.PKT_ID offset 2-804
?WIN_GINT.RSV offset 2-804
?WIN_GTITLE.PKT_ID offset 2-807
?WIN_GTITLE_PKTID value 2-807
?WIN_GTITLE.TITLE offset 2-807
?WIN_GTITLE.TITLE_BUF_LEN offset 2-807
?WIN_GTITLE_TITLE_LEN 2-806
?WIN_HIDE_GROUP function code 2-800, 2-788
?WIN_HIDE_WINDOW function code 2-800, 2-788
?WIN_LANG_ID_CANADA_ENGLISH value 2-814
?WIN_LANG_ID_CANADA_FRENCH value 2-814
?WIN_LANG_ID_DANISH_NORWEGIAN value 2-814
?WIN_LANG_ID_FRENCH value 2-814
?WIN_LANG_ID_GERMAN value 2-814
?WIN_LANG_ID_ITALIAN value 2-814
?WIN_LANG_ID_KATAKANA value 2-814
?WIN_LANG_ID_SPANISH value 2-814
?WIN_LANG_ID_SWEDISH_FINNISH value 2-814
?WIN_LANG_ID_SWISS_GERMAN value 2-814
?WIN_LANG_ID_UK value 2-814
?WIN_LANG_ID_US value 2-814
?WIN_OUTBACK_GROUP function code 2-799, 2-788
?WIN_OUTBACK_WINDOW function code 2-799, 2-788
?WIN_PALETTE_TYPE_D460 value 2-796, 2-812
?WIN_PALETTE_TYPE_DEFAULT value 2-795, 2-812
?WIN_PALETTE_TYPE_MONO_D460 value 2-796, 2-812
?WIN_PALETTE_TYPE_SHARED value 2-796, 2-812
?WIN_PALETTE_TYPE_SYSTEM value 2-796, 2-812
?WIN_PALETTE_TYPE_TRUE value 2-795, 2-812
?WIN_PALETTE_TYPE_USER value 2-795, 2-812
?WIN_PERMANENCE_OFF function code 2-802, 2-789
?WIN_PERMANENCE_ON function code 2-802, 2-789
?WIN_PKT.CHAN_NUM offset 2-790f
?WIN_PKT.FLAGS offset 2-790f
?WIN_PKT.FLAGS.IN_CHAN value 2-791
?WIN_PKT.FLAGS.IN_PATH value 2-791
?WIN_PKT.FLAGS.IN_WIND_ID value 2-791
?WIN_PKT.FUNC offset 2-790f, 2-787
?WIN_PKT_LEN value 2-790
?WIN_PKT.PATH offset 2-790f, 2-807
?WIN_PKT.PATH_BUF_LEN offset 2-790f
?WIN_PKT.PATH_LEN offset 2-790f, 2-807
?WIN_PKT.PKT_ID offset 2-790f
?WIN_PKT.SUBPKT offset 2-790f, 2-796, 2-799ff, 2-802, 2-807, 2-813
?WIN_PKT.WIND_ID offset 2-790f, 2-813
?WIN_PTR_TYPE_ABSOLUTE value 2-814
?WIN_PTR_TYPE_RELATIVE value 2-814
?WIN_RDPAL.ARRAY_PKT_ID offset 2-819
?WIN_RDPAL_ARRAY_PKTID value 2-819
?WIN_RDPAL.ENTRY_CNT offset 2-819
?WIN_RDPAL.FIRST_PAL_INDEX offset 2-819
?WIN_RDPAL_LEN value 2-819
?WIN_RDPAL.PH0_ARRAY offset 2-819
?WIN_RDPAL.PH1_ARRAY offset 2-819

Licensed Material - Property of Data General Corporation

?WIN_RDPAL.PKT_ID offset 2-819
?WIN_RDPAL_PKTID value 2-819
?WIN_READ_PALETTE function code 2-818f, 2-789
?WIN_RETURN_DEVICE_WINDOWS function code 2-816, 2-789
?WIN_RETURN_GROUP_WINDOWS function code 2-815, 2-789
?WIN_RTN_DEVWINDS.ARRAY_PKT_ID offset 2-816
?WIN_RTN_DEVWINDS.I_NUM_WINDS offset 2-816
?WIN_RTN_DEVWINDS_LEN value 2-816
?WIN_RTN_DEVWINDS.O_NUM_WINDS offset 2-816
?WIN_RTN_DEVWINDS.PKT_ID offset 2-816
?WIN_RTN_DEVWINDS_PKTID value 2-816
?WIN_RTN_DEVWINDS.WIND_BUF offset 2-816
?WIN_RTN_DEVWINDS_WIND_BUF_PKTID value 2-816
?WIN_RTN_GRPWINDS.ARRAY_PKT_ID offset 2-815
?WIN_RTN_GRPWINDS.I_NUM_WINDS offset 2-815
?WIN_RTN_GRPWINDS_LEN value 2-815
?WIN_RTN_GRPWINDS.O_NUM_WINDS offset 2-815
?WIN_RTN_GRPWINDS.PKT_ID offset 2-815
?WIN_RTN_GRPWINDS_PKTID value 2-815
?WIN_RTN_GRPWINDS.WIND_BUF offset 2-815
?WIN_RTN_GRPWINDS_WIND_BUF_PKTID value 2-815
?WIN_SET_TITLE function code 2-806, 2-789
?WIN_SET_USER_INTERFACE function code 2-802ff, 2-789
?WIN_SINT.BORDER_TYPE offset 2-803f
?WIN_SINT.FLAGS offset 2-803f
?WIN_SINT.FLAGS.ELEVATORS value 2-803
?WIN_SINT.FLAGS.FULL_REV value 2-803
?WIN_SINT.FLAGS.HELP value 2-803
?WIN_SINT.FLAGS.MOVE_X value 2-803
?WIN_SINT.FLAGS.MOVE_Y value 2-803
?WIN_SINT.FLAGS.SCROLL_X value 2-803
?WIN_SINT.FLAGS.SCROLL_Y value 2-803
?WIN_SINT.FLAGS.SZ_X value 2-803
?WIN_SINT.FLAGS.SZ_Y value 2-803
?WIN_SINT.FLAGS.WIRE_BRDRS value 2-803
?WIN_SINT_LEN value 2-803
?WIN_SINT.PKT_ID offset 2-803
?WIN_SINT.RSV offset 2-803
?WIN_STATUS.FLAGS offset 2-809f
?WIN_STATUS.FLAGS.GRP_HIDDEN value 2-810
?WIN_STATUS.FLAGS.GRP_SUSPENDED value 2-810
?WIN_STATUS.FLAGS.KEYB_ENABLED value 2-810
?WIN_STATUS.FLAGS.PERM_ON value 2-810
?WIN_STATUS.FLAGS.WIND_HIDDEN value 2-810
?WIN_STATUS.FLAGS.WIND_SUSPENDED value 2-810
?WIN_STATUS.INPUT_BUF_SZ offset 2-809, 2-811
?WIN_STATUS_LEN_REV1 value 2-809
?WIN_STATUS.PAL_TYPE offset 2-809, 2-812
?WIN_STATUS.PIXEL_DEPTH offset 2-809, 2-811
?WIN_STATUS.PKT_ID offset 2-809f
?WIN_STATUS_PKTID value 2-810
?WIN_STATUS.SCAN_ORG.X offset 2-809f
?WIN_STATUS.SCAN_ORG.Y offset 2-809f
?WIN_STATUS.SCAN_SZ.X offset 2-809f
?WIN_STATUS.SCAN_SZ.Y offset 2-809f
?WIN_STATUS.VIEW_ORG.X offset 2-809f
?WIN_STATUS.VIEW_ORG.Y offset 2-809f
?WIN_STATUS.VIEW_SZ.X offset 2-809f
?WIN_STATUS.VIEW_SZ.Y offset 2-809f
?WIN_STATUS.VT_SZ.X offset 2-809, 2-811
?WIN_STATUS.VT_SZ.Y offset 2-809, 2-811
?WIN_STATUS.VT_TYPE offset 2-809, 2-811
?WIN_STATUS.WIND_ORG.X offset 2-809, 2-811
?WIN_STATUS.WIND_ORG.Y offset 2-809, 2-811
?WIN_STATUS.WIND_SZ.X offset 2-809, 2-811
?WIN_STATUS.WIND_SZ.Y offset 2-809, 2-811
?WIN_STITLE_LEN value 2-806
?WIN_STITLE.PKT_ID offset 2-806
?WIN_STITLE_PKTID value 2-806
?WIN_STITLE.RSV offset 2-806
?WIN_STITLE.TITLE offset 2-806
?WIN_STITLE.TITLE_BUF_LEN offset 2-806
?WIN_SUSPEND_GROUP function code 2-801, 2-788
?WIN_SUSPEND_WINDOW function code 2-801, 2-788
?WIN_UNHIDE_GROUP function code 2-800, 2-788
?WIN_UNHIDE_WINDOW function code 2-800, 2-788
?WIN_UNSUSPEND_GROUP function code 2-801, 2-788
?WIN_UNSUSPEND_WINDOW function code 2-801, 2-788
?WIN_UPFRONT_GROUP function code 2-799, 2-788
?WIN_UPFRONT_WINDOW function code 2-799, 2-788
?WIN_VT_TYPE_D460 value 2-794, 2-811

?WIN_VT_TYPE_DEFAULT value 2-794
?WIN_VT_TYPE_MONO_D460 value 2-794, 2-811
?WIN_VT_TYPE_PIXMAP value 2-794, 2-811
?WIN_WINDOW_STATUS function code 2-808ff, 2-789
?WIN_WRITE_PALETTE function code 2-817f, 2-789
?WIN_WRPAL.ARRAY_PKT_ID offset 2-818
?WIN_WRPAL_ARRAY_PKTID value 2-818
?WIN_WRPAL.ENTRY_CNT offset 2-818
?WIN_WRPAL.FIRST_PAL_INDEX offset 2-818
?WIN_WRPAL_LEN value 2-818
?WIN_WRPAL.PH0_ARRAY offset 2-818
?WIN_WRPAL.PH1_ARRAY offset 2-818
?WIN_WRPAL.PKT_ID offset 2-818
?WIN_WRPAL_PKTID value 2-818
?WIRE system call 2-820f
.WORD MASM pseudo-operation 1-5
?WRAC value 2-773
?WRB system call 2-586ff, 2-822
?WRITE system call 2-595ff, 2-823
  example of, *see* program BOOMER.SR
  example of, *see* program DLIST.SR
  example of, *see* program FILCREATE.SR
  example of, *see* program HEAR.SR
  example of, *see* program INRING.SR
  example of, *see* program NEWTASK.SR
  example of, *see* program RINGLOAD.SR
  example of, *see* program RUNTIME.SR
  example of, *see* program WRITE.SR
WRITE.SR sample program B-23ff, B-2
writing to a palette 2-229f
writing to a window's palette 2-817f
?WRUDA system call 2-593f, 2-824
?WTSIG system call 2-825f
?WVBPL value 2-775
?WVWPL value 2-775

X

?X2AP value 2-129
?X2CD value 2-128
?X2CM value 2-129
?X2CN value 2-128
?X2CP value 2-129
?X2DD value 2-129
?X2EB value 2-128
?X2EX value 2-128
?X2HO value 2-128
?X2LN value 2-128
?X2RC value 2-129
?X2RM value 2-129
?X2SD value 2-129
?X2SE value 2-128
?X2TO value 2-129
?X3CO value 2-129
?X3TR value 2-129
?XAFD offset 2-123, 2-126, 2-138, 2-140, 2-142
?XAFT offset 2-123, 2-126, 2-138, 2-140, 2-142
?XBFXR value 2-834
?XBORD value 2-834
?XD2FG offset 2-145f
?XD3FG offset 2-145f
?XDAD offset 2-146
?XDAT offset 2-123f, 2-138, 2-140
?XDBP offset 2-146
?XDCOP offset 2-146
?XDEP offset 2-146
?XDJN offset 2-146
?XDLMT offset 2-146
?XDPN offset 2-146
?XDQP offset 2-146
?XDRSV offset 2-146
?XDSD offset 2-146
?XDSFG offset 2-145f
?XDSQN offset 2-146
?XDST offset 2-146
?XDTA offset 2-146
?XDUL offset 2-121
?XDUN offset 2-146
?XDUT offset 2-121
?XFBI value 2-125, 2-142
?XFBP offset 2-123, 2-126, 2-138, 2-140, 2-142
?XFCAN code 2-133f
?XFCAN value 2-132
?XFDA value 2-125, 2-142
?XFDUN code 2-121
?XFFO value 2-126, 2-142
?XFFTA value 2-124
?XFG2 offset 2-138, 2-140, 2-145f
?XFGRT value 2-834
?XFGS offset 2-123, 2-125f, 2-138, 2-140, 2-142, 2-145f
?XFGWD value 2-834
?XFHAM value 2-124
?XFHK offset 2-143ff, 2-146
?XFHOL code 2-133f
?XFHOL value 2-132
?XFLG offset 2-144
?XFLK offset 2-143ff, 2-146
?XFLPT value 2-124
?XFMLT code 2-120, 2-138
?XFMNT code 2-138
?XFMOD code 2-140f, 2-452
?XFMUN code 2-119, 2-138
?XFNO value 2-125, 2-142
?XFNQN code 2-143f, 2-144
?XFNR value 2-125, 2-142
?XFOP value 2-125, 2-142

Licensed Material - Property of Data General Corporation

?XFP1 offset 2-131ff, 2-134ff, 2-137
?XFP2 offset 2-131ff, 2-134ff, 2-137
?XFP2L offset 2-131ff, 2-134ff
?XFP3 offset 2-136f
?XFP4 offset 2-133f, 2-136f
?XFPE value 2-125, 2-138
?XFPLT value 2-124
?XFQDS code 2-145f
?XFQN offset 2-145f
?XFQT offset 2-144
?XFRA value 2-126
?XFSH value 2-125, 2-142
?XFSNA value 2-124
?XFSTS code 2-135
?XFSUB code 2-138
?XFSUB value 2-124
?XFTI value 2-126, 2-142
?XFUNH code 2-133f
?XFUNH value 2-132
?XFWP offset 2-144, 2-146
?XFXDU code 2-139
?XFXML flags 2-121
?XFXTS code 2-136f
?XFXUN code 2-119
?XFXUN flags 2-121
?XHBP offset 2-140
?XICNT offset 2-833, 2-835
?XID1 offset 2-833, 2-835
?XID2 offset 2-833, 2-835
?XID3 offset 2-833, 2-835
?XIDIR offset 2-833f
?XIFUN offset 2-833f
?XIGTD value 2-834
?XII1 value 2-834
?XII2 value 2-834
?XILDN offset 2-833f
?XILN value 2-833
?XINIT system call 2-827ff
?XINIT_FIXUP_REC value 2-830
?XINIT_MAXIMAGES value 2-830, 2-835
?XINIT_OVERRIDE value 2-830
?XINIT_PKT.COUNT offset 2-829f
?XINIT_PKT.DIR offset 2-829f
?XINIT_PKT.FUNC offset 2-829f
?XINIT_PKT.LDID1 offset 2-829f
?XINIT_PKT.LDID2 offset 2-829, 2-831
?XINIT_PKT.LDID3 offset 2-829, 2-831
?XINIT_PKT.LDNAME offset 2-829f
?XINIT_PKT_LEN value 2-829
?XINIT_PKT.OPTIONS offset 2-829f
?XINIT_PKT.PKT_ID offset 2-829f
?XINIT_PKT_PKTID value 2-830
?XINIT_PKT.PUL1 offset 2-829, 2-831
?XINIT_PKT.PUL2 offset 2-829, 2-831
?XINIT_PKT.PUL3 offset 2-829, 2-831
?XINIT_PKT.RES offset 2-829f
?XINIT_PUL_PKTID value 2-831
?XINIT_ROOT value 2-830
?XINIT_TARGET_DIR value 2-830
?XINIT_WORKING_DIR value 2-830
?XIOP offset 2-833f
?XIP1 offset 2-833, 2-835
?XIP2 offset 2-833, 2-835
?XIP3 offset 2-833, 2-835
?XIPHI offset 2-833f
?XIPLO offset 2-833f
?XIR1 offset 2-833f
?XIR2 offset 2-833f
?XIR3 offset 2-833f
?XIR4 offset 2-833f
?XIR5 offset 2-833, 2-835
?XIR6 offset 2-833, 2-835
?XIR7 offset 2-833, 2-835
?XIR8 offset 2-833, 2-835
?XIR9 offset 2-833, 2-835
?XIRA offset 2-833, 2-835
?XIRES offset 2-833f
?XL1CAN value 2-133
?XL1HOL value 2-133
?XL1UNH value 2-133
?XLCAN value 2-131
?XLDUN value 2-121
?XLFWP value 2-146
?XLHOL value 2-131
?XLMLT value 2-120
?XLMNT value 2-138
?XLMOD value 2-140
?XLMT offset 2-123ff, 2-138, 2-140, 2-142
?XLMUN value 2-119
?XLNQN value 2-144
?XLQNB value 2-144
?XLSTS value 2-135
?XLTH value 2-123
?XLUNH value 2-131
?XLXTS value 2-136
?XMBP offset 2-140
?XMEL value 2-121
?XMFC value 2-121
?XMFI value 2-121
?XMFR value 2-121
?XMIBM value 2-42
?XMLE offset 2-120
?XMLF offset 2-120
?XMLL offset 2-120
?XMLR offset 2-120
?XMLS offset 2-120
?XMLT offset 2-120
?XMLV offset 2-120
?XMLX value 2-120
?XMT system call 2-837
?XMTW system call 2-838
example of, *see* program BOOMER.SR

?XMUE offset 2-119
?XMUF offset 2-119
?XMUL offset 2-119
?XMUQ offset 2-119
?XMUR offset 2-119
?XMUS offset 2-119
?XMUT offset 2-119, 2-121
?XMUX value 2-119
?XNRQ offset 2-144, 2-146
?XNRT offset 2-144, 2-146
?XPBP offset 2-123, 2-125f, 2-138, 2-140
?XPCH offset 2-841, 2-845
?XPCID offset 2-842, 2-846
?XPCPL offset 2-841, 2-845
?XPCPU offset 2-842, 2-846
?XPCW offset 2-841, 2-845
?XPDBS offset 2-841, 2-846
?XPDIS offset 2-841, 2-846, 2-848
?XPDRS offset 2-841, 2-846
?XPEX offset 2-841, 2-845
?XPF2 offset 2-841, 2-844
?XPF3 offset 2-841, 2-845
?XPF4 offset 2-841, 2-845
?XPF5 offset 2-841, 2-845
?XPFA offset 2-841, 2-846
?XPFL offset 2-841, 2-844
?XPFP offset 2-841, 2-844
?XPGBS offset 2-842, 2-847
?XPGRS offset 2-842, 2-847
?XPIH offset 2-841, 2-846
?XPLFA offset 2-842, 2-846
?XPLL offset 2-842, 2-846
?XPLTH value 2-842
?XPMX offset 2-841, 2-846
?XPNBS offset 2-842, 2-847
?XPNR offset 2-841, 2-844
?XPNRS offset 2-842, 2-847
?XPPD offset 2-841, 2-845
?XPPG offset 2-842, 2-846
?XPPH offset 2-841, 2-845
?XPPL offset 2-842, 2-846
?XPPR offset 2-841, 2-845
?XPPU offset 2-842, 2-847
?XPPV offset 2-841, 2-845
?XPR1 offset 2-841, 2-845
?XPRH offset 2-841, 2-845
?XPRI offset 2-123, 2-125, 2-138, 2-140, 2-142
?XPRV offset 2-118ff
?XPSF offset 2-841, 2-844
?XPSID value 2-844
?XPSL offset 2-842, 2-846
?XPSNS offset 2-841, 2-844
?XPSP offset 2-841, 2-844
?XPSQF offset 2-841, 2-844
?XPSTAT system call 2-839ff
?XPSW offset 2-841, 2-844
?XPUBS offset 2-842, 2-847
?XPULC offset 2-842, 2-846
?XPUN offset 2-842, 2-847
?XPUPD offset 2-842, 2-847
?XPURS offset 2-842, 2-847
?XPWD offset 2-140
?XPWM offset 2-841, 2-846
?XPWS offset 2-841, 2-846
?XQ1FG offset 2-144
?XQNJ offset 2-144
?XQQN offset 2-144
?XQQT offset 2-144
?XQRSV offset 2-144
?XRES1 offset 2-123f
?XRES2 offset 2-123f, 2-138, 2-140
?XRFNC offset 2-118ff
?XSEQ offset 2-123, 2-126, 2-138, 2-140f
?XT16T value 2-752, 2-756
?XT32T value 2-752, 2-756
?XTABR value 2-752, 2-756
?XTAOS value 2-752, 2-756
?XTBCX value 2-752, 2-756
?XTCCX value 2-752, 2-756
?XTCIN value 2-752, 2-756
?XTIM offset 2-123f, 2-138, 2-140
?XTR16 value 2-752, 2-756
?XTR32 value 2-752, 2-756
?XTSUP value 2-752, 2-756
?XTYP offset 2-123f, 2-138, 2-140f
?XUSR offset 2-140
XW0 value 2-140
XW1 value 2-140
XW2 value 2-140
XW3 value 2-140
?XWM0 offset 2-140, 2-142
?XWM1 offset 2-140, 2-142
?XWM2 offset 2-140, 2-142
?XWM3 offset 2-140, 2-142
?XWW0 offset 2-138, 2-140, 2-142
?XWW0L offset 2-138, 2-140
?XWW1 offset 2-138, 2-140, 2-142
?XWW1L offset 2-138, 2-140
?XWW2 offset 2-138, 2-140, 2-142
?XWW2L offset 2-138, 2-140
?XWW3 offset 2-138, 2-140, 2-142
?XWW3L offset 2-138, 2-140
?XXDL offset 2-139
?XXDP offset 2-139
?XXDQ offset 2-139
?XXDT offset 2-139
?XXW0 offset 2-123, 2-127
?XXW0L offset 2-123, 2-127
?XXW1 offset 2-123, 2-128
?XXW1L offset 2-123, 2-128
?XXW2 offset 2-123, 2-128f
?XXW2L offset 2-123, 2-129

Licensed Material - Property of Data General Corporation

?XXW3 offset 2-123, 2-129
?XXW3L offset 2-123, 2-129

Z

?ZAC value 2-415
?ZAC3 offset 2-418
?ZAC4 offset 2-418
?ZACA offset 2-418
?ZACF offset 2-418
?ZACI offset 2-418
?ZACR offset 2-418
?ZACZ value 2-418
?ZAG value 2-415, 2-419
?ZAGF offset 2-419
?ZAGI offset 2-419
?ZAGL value 2-419
?ZAGN offset 2-419
?ZAGZ value 2-419
?ZAK value 2-416f
?ZAKA offset 2-491
?ZAKB offset 2-491
?ZAKF offset 2-491
?ZAKI offset 2-491
?ZAKL value 2-491
?ZAKR offset 2-491
?ZAKZ value 2-491
?ZAL value 2-415
?ZBI value 2-415
?ZBIA offset 2-423
?ZBIB offset 2-423
?ZBIF offset 2-423
?ZBIF value 2-423
?ZBII offset 2-423
?ZBIL value 2-423
?ZBIZ value 2-423
?ZBL value 2-415
?ZBLA offset 2-421
?ZBLB offset 2-421
?ZBLF offset 2-421
?ZBLI offset 2-421
?ZBLL value 2-421
?ZBLZ value 2-421
?ZBO value 2-415
?ZBOA offset 2-422
?ZBOB offset 2-422
?ZBOF offset 2-422
?ZBOI offset 2-422
?ZBOL value 2-422
?ZBOZ value 2-422
?ZBR value 2-415
?ZBRF offset 2-424
?ZBRI offset 2-424
?ZBRL value 2-424
?ZBRS offset 2-424
?ZBRZ value 2-424
?ZCA value 2-415
?ZCAI offset 2-425
?ZCAL value 2-425
?ZCAR offset 2-425
?ZCAS offset 2-425
?ZCAZ value 2-425
?ZCL value 2-415
?ZCO value 2-415
?ZCOF offset 2-429
?ZCOI offset 2-429
?ZCOL value 2-429
?ZCOS offset 2-429
?ZCP value 2-415
?ZCPI offset 2-430
?ZCPL value 2-430
?ZCPN offset 2-430
?ZCPR offset 2-430
?ZCPZ value 2-430
?ZCR value 2-415
?ZCRF offset 2-431
?ZCRI offset 2-431
?ZCRL value 2-431
?ZCRN offset 2-431
?ZCRQ offset 2-431
?ZCRR offset 2-431
?ZCRZ value 2-431
?ZCS value 2-415
?ZCS3 offset 2-427f
?ZCSA offset 2-427f
?ZCSB offset 2-427f
?ZCSF offset 2-427f
?ZCSI offset 2-427f
?ZCSK offset 2-427f
?ZCSL value 2-427
?ZCSN offset 2-427f
?ZCSP offset 2-427f
?ZCSR offset 2-427f
?ZCSU offset 2-427f
?ZCSZ value 2-428
?ZDE value 2-415
?ZDF value 2-415
?ZDFA offset 2-432
?ZDFB offset 2-432
?ZDFF offset 2-432
?ZDFI offset 2-432
?ZDFL value 2-432
?ZDFZ value 2-432
?ZDG value 2-416
?ZDI value 2-415
?ZDIF offset 2-434
?ZDII offset 2-434
?ZDIL value 2-434
?ZDIN offset 2-434
?ZDIZ value 2-434

?ZDS value 2-415
?ZDSF offset 2-435
?ZDSI offset 2-435
?ZDSL value 2-435
?ZDSM offset 2-435
?ZDSZ value 2-435
?ZEL value 2-415
?ZELF offset 2-436
?ZELI offset 2-436
?ZELN value 2-436
?ZELR offset 2-436
?ZELZ value 2-436
?ZEN value 2-415
?ZEN3 offset 2-437f
?ZENC offset 2-437f
?ZENF offset 2-437f
?ZENI offset 2-437f
?ZENK offset 2-437f
?ZENL value 2-437
?ZENR offset 2-437f
?ZENT offset 2-437f
?ZENZ value 2-438
?ZEV value 2-415
?ZEVF offset 2-439
?ZEVI offset 2-439
?ZEVL value 2-439
?ZEVR offset 2-439
?ZEVZ value 2-439
?ZFL value 2-415
?ZFLF offset 2-440
?ZFLI offset 2-440
?ZFLL value 2-440
?ZFLS offset 2-440
?ZFLZ value 2-440
?ZFN value 2-415
?ZFO value 2-415
?ZFOA offset 2-442
?ZFOB offset 2-442
?ZFOF offset 2-442
?ZFOI offset 2-442
?ZFOL value 2-442
?ZFOZ value 2-442
?ZHA value 2-415, 2-417
?ZHAF offset 2-443
?ZHAI offset 2-443
?ZHAL value 2-443
?ZHAR offset 2-443
?ZHAZ value 2-443
?ZHE value 2-415
?ZHEH offset 2-444
?ZHEI offset 2-444
?ZHEL value 2-444
?ZHER offset 2-444
?ZHEZ value 2-444
?ZHO value 2-415
?ZHOI offset 2-445
?ZHOL value 2-445
?ZHOR offset 2-445
?ZHOS offset 2-445
?ZHOZ value 2-445
?ZJ22 value 2-455
?ZJ30 value 2-458
?ZJ90 value 2-462
?ZJAF value 2-491
?ZJAG value 2-491
?ZJAH value 2-491
?ZJF1 value 2-423
?ZJH0 value 2-436
?ZJI0 value 2-439
?ZJS1 value 2-465
?ZJV1 value 2-478
?ZJV5 value 2-478
?ZJV6 value 2-478
?ZJVD value 2-478
?ZJVE value 2-478
?ZLI value 2-415
?ZLIA offset 2-446
?ZLIB offset 2-446
?ZLII offset 2-446
?ZLIL value 2-446
?ZLIZ value 2-446
?ZLO value 2-415
?ZLO3 offset 2-447f
?ZLO4 offset 2-447f
?ZLO5 offset 2-447f
?ZLO6 offset 2-447f
?ZLOC offset 2-447f
?ZLOF offset 2-447f
?ZLOI offset 2-447f
?ZLOL value 2-447
?ZLOP offset 2-447f
?ZLOZ value 2-448
?ZLP value 2-415
?ZLPI offset 2-449
?ZLPL value 2-449
?ZLPN offset 2-449
?ZLPR offset 2-449
?ZLPZ value 2-449
?ZMD value 2-415, 2-417
?ZME value 2-415
?ZMO value 2-415
?ZMOA offset 2-453
?ZMOB offset 2-453
?ZMOF offset 2-453
?ZMOI offset 2-453
?ZMOL value 2-453
?ZMOM offset 2-453
?ZMOR offset 2-453
?ZMOZ value 2-453
?ZMP value 2-415
?ZMS value 2-415
?ZMS0 offset 2-454, 2-456

Licensed Material - Property of Data General Corporation

?ZMS3 offset 2-454, 2-456
?ZMS4 offset 2-454, 2-456
?ZMS5 offset 2-454, 2-456
?ZMS6 offset 2-454, 2-456
?ZMS7 offset 2-454, 2-456
?ZMS8 offset 2-454, 2-456
?ZMS9 offset 2-454, 2-456
?ZMSA offset 2-454, 2-456
?ZMSD offset 2-454, 2-456
?ZMSE offset 2-454f
?ZMSF offset 2-454f
?ZMSG offset 2-454f
?ZMSI offset 2-454f
?ZMSK offset 2-454f
?ZMSL value 2-454
?ZMSM offset 2-454, 2-456
?ZMSP offset 2-454f
?ZMSQ value 2-455
?ZMSR offset 2-454, 2-456
?ZMSS offset 2-454, 2-456
?ZMST offset 2-454, 2-456
?ZMSU offset 2-454, 2-456
?ZMSV offset 2-454, 2-456
?ZMSX offset 2-454, 2-456
?ZMSZ offset 2-454, 2-456
?ZON value 2-415
?ZOP value 2-415
?ZOPF offset 2-458
?ZOPI offset 2-458
?ZOPL value 2-458
?ZOPR offset 2-458
?ZOPZ value 2-458
?ZPA value 2-415
?ZPAF offset 2-459
?ZPAI offset 2-459
?ZPAL value 2-459
?ZPAS offset 2-459
?ZPAZ value 2-459
?ZPE value 2-415
?ZPE3 offset 2-460
?ZPE4 offset 2-460
?ZPEF offset 2-460
?ZPEI offset 2-460
?ZPEL value 2-460
?ZPER offset 2-460
?ZPEU offset 2-460
?ZPEV offset 2-460
?ZPEZ value 2-460
?ZPI value 2-415
?ZPIA offset 2-461
?ZPIB offset 2-461
?ZPIF offset 2-461
?ZPII offset 2-461
?ZPIL value 2-461
?ZPIR offset 2-461
?ZPIZ value 2-461
?ZPR value 2-415, 2-417
?ZPRF offset 2-462
?ZPRI offset 2-462
?ZPRL value 2-462
?ZPRR offset 2-462
?ZPRZ value 2-462
?ZPU value 2-415
?ZQP value 2-415
?ZQPA offset 2-464f
?ZQPB offset 2-464f
?ZQPC offset 2-464f
?ZQPF offset 2-464f
?ZQPH offset 2-464
?ZQPI offset 2-464f
?ZQPL value 2-464
?ZQPN offset 2-464f
?ZQPO offset 2-464
?ZQPP offset 2-464f
?ZQPS offset 2-464
?ZQPU offset 2-464f
?ZQPX value 2-464
?ZQPZ value 2-465
?ZQSS value 2-478
?ZRE value 2-416
?ZREF offset 2-467
?ZREI offset 2-467
?ZREL value 2-467
?ZRER offset 2-467
?ZREZ value 2-467
?ZRF value 2-416f
?ZRFF offset 2-466
?ZRFI offset 2-466
?ZRFL value 2-466
?ZRFM offset 2-466
?ZRFZ value 2-466
?ZRG value 2-416
?ZRG3 offset 2-492
?ZRGF offset 2-492
?ZRGI offset 2-492
?ZRGL value 2-492
?ZRGQ offset 2-492
?ZRGZ value 2-492
?ZRS value 2-416
?ZRS3 offset 2-494f
?ZRSA offset 2-494f
?ZRSB offset 2-494f
?ZRSF offset 2-494f
?ZRSI offset 2-494f
?ZRSK offset 2-494f
?ZRSL value 2-494
?ZRSQ offset 2-494f
?ZRSR offset 2-494f
?ZRSZ value 2-495
?ZRT value 2-416
?ZRTA offset 2-468
?ZRTB offset 2-468

?ZRTI offset 2-468
?ZRTL value 2-468
?ZRTZ value 2-468
?ZSI value 2-416
?ZSIF offset 2-469
?ZSII offset 2-469
?ZSIL value 2-469
?ZSIS offset 2-469
?ZSIZ value 2-469
?ZSK value 2-416
?ZSK3 offset 2-473
?ZSKF offset 2-473
?ZSKI offset 2-473
?ZSKL value 2-473
?ZSKR offset 2-473
?ZSKZ value 2-473
?ZSP value 2-416
?ZSP3 offset 2-470, 2-472
?ZSP4 offset 2-470, 2-472
?ZSP5 offset 2-470, 2-472
?ZSP6 offset 2-470, 2-472
?ZSP7 offset 2-470, 2-472
?ZSP8 offset 2-470, 2-472
?ZSP9 offset 2-470, 2-472
?ZSPB offset 2-470f
?ZSPC offset 2-470f
?ZSPD offset 2-470, 2-472
?ZSPE offset 2-470, 2-472
?ZSPF offset 2-470f
?ZSPH offset 2-470f
?ZSPI offset 2-470f
?ZSPK offset 2-470, 2-472
?ZSPL value 2-470
?ZSPN offset 2-470f
?ZSPP offset 2-470f
?ZSPQ offset 2-470, 2-472
?ZSPR offset 2-470, 2-472
?ZSPS offset 2-470, 2-472
?ZSPT offset 2-470f
?ZSPV offset 2-470f
?ZSPZ value 2-471
?ZSR value 2-416
?ZSR3 offset 2-474
?ZSR4 offset 2-474
?ZSR5 offset 2-474
?ZSR6 offset 2-474
?ZSR7 offset 2-474
?ZSRB offset 2-474
?ZSRC offset 2-474
?ZSRF offset 2-474
?ZSRI offset 2-474
?ZSRL value 2-474
?ZSRM offset 2-474
?ZSRQ offset 2-474
?ZSRR offset 2-474
?ZSRS offset 2-474
?ZSRX offset 2-474
?ZSRZ value 2-475
?ZSS value 2-416
?ZSS0 offset 2-477, 2-480
?ZSS5 offset 2-477, 2-479
?ZSS6 offset 2-477, 2-479
?ZSS7 offset 2-477, 2-479
?ZSS8 offset 2-477, 2-479
?ZSS9 offset 2-477, 2-479
?ZSSA offset 2-477, 2-480
?ZSSB offset 2-477, 2-479
?ZSSC offset 2-477, 2-479
?ZSSD offset 2-477, 2-479
?ZSSE offset 2-477, 2-479
?ZSSF offset 2-477f
?ZSSG offset 2-477, 2-479
?ZSSH offset 2-477, 2-480
?ZSSI offset 2-477f
?ZSSJ offset 2-477, 2-480
?ZSSK offset 2-477, 2-479
?ZSSL value 2-477
?ZSSM offset 2-477, 2-480
?ZSSN offset 2-477, 2-479
?ZSSO offset 2-477, 2-479
?ZSSP offset 2-477, 2-479
?ZSSR offset 2-477, 2-479
?ZSSS offset 2-477, 2-479
?ZSST offset 2-477, 2-480
?ZSSU offset 2-477, 2-479
?ZSSV offset 2-477, 2-479
?ZSSX offset 2-477, 2-479
?ZSSY offset 2-477, 2-479
?ZSSZ offset 2-477, 2-480
?ZST value 2-416
?ZSTA offset 2-481
?ZSTB offset 2-481
?ZSTF offset 2-481
?ZSTI offset 2-481
?ZSTL value 2-481
?ZSTZ value 2-481
?ZSX0 offset 2-477, 2-479
?ZTE value 2-416
?ZTR value 2-416
?ZTRI offset 2-483
?ZTRL value 2-483
?ZTRR offset 2-483
?ZTRT offset 2-483
?ZTRZ value 2-483
?ZUH value 2-416
?ZUHI offset 2-484
?ZUHL value 2-484
?ZUHR offset 2-484
?ZUHS offset 2-484
?ZUHZ value 2-484
?ZUL value 2-416
?ZULF offset 2-487

Licensed Material - Property of Data General Corporation

?ZULI offset 2-487
?ZULL value 2-487
?ZULS offset 2-487
?ZULZ value 2-487
?ZUN value 2-416
?ZUNF offset 2-488
?ZUNI offset 2-488
?ZUNL value 2-488
?ZUNS offset 2-488
?ZUNZ value 2-488
?ZUS value 2-416
?ZUS3 offset 2-485f
?ZUS4 offset 2-485f
?ZUS5 offset 2-485f
?ZUS6 offset 2-485f
?ZUS7 offset 2-485f
?ZUSF offset 2-485f
?ZUSI offset 2-485f
?ZUSK offset 2-485f
?ZUSL value 2-485
?ZUSM offset 2-485f
?ZUSP offset 2-485f
?ZUSR offset 2-485f
?ZUSU offset 2-485f
?ZUSV offset 2-485f
?ZUSZ value 2-486
?ZVE value 2-416
?ZVEF offset 2-489
?ZVEI offset 2-489
?ZVEL value 2-489
?ZVES offset 2-489
?ZVEZ value 2-489
?ZXB value 2-416
?ZXBA offset 2-490
?ZXBB offset 2-490
?ZXBI offset 2-490
?ZXBL value 2-490
?ZXBZ value 2-490
?ZXFG offset 2-414, 2-417
?ZXFU offset 2-414, 2-415f
?ZXID offset 2-414f
?ZXIZ value 2-415
?ZXLN value 2-414
?ZXNA offset 2-414, 2-417
?ZXNB offset 2-414, 2-417
?ZXNL offset 2-414, 2-417
?ZXRS offset 2-414, 2-417
?ZXSP offset 2-414, 2-417
?ZXTP offset 2-414, 2-417
?ZY00 value 2-435
?ZY10 value 2-453
?ZY20 value 2-455
?ZY21 value 2-455
?ZY22 value 2-455
?ZY23 value 2-455
?ZY24 value 2-455
?ZY25 value 2-455
?ZY26 value 2-455
?ZY27 value 2-455
?ZY28 value 2-455
?ZY29 value 2-455
?ZY2G value 2-455
?ZY2H value 2-455
?ZY2I value 2-455
?ZY2J value 2-455
?ZY30 value 2-458
?ZY40 value 2-460
?ZY50 value 2-466
?ZY60 value 2-467
?ZY70 value 2-486
?ZY71 value 2-486
?ZY72 value 2-486
?ZY73 value 2-486
?ZY74 value 2-486
?ZY80 value 2-448
?ZY81 value 2-448
?ZY82 value 2-448
?ZY83 value 2-448
?ZY84 value 2-448
?ZY85 value 2-448
?ZY86 value 2-448
?ZY87 value 2-448
?ZY90 value 2-462
?ZY9G value 2-418
?ZY9H value 2-418
?ZY9I value 2-418
?ZYAF value 2-491
?ZYAG value 2-491
?ZYAH value 2-491
?ZYB0 value 2-428
?ZYB1 value 2-428
?ZYB2 value 2-428
?ZYB3 value 2-428
?ZYB4 value 2-428
?ZYB5 value 2-428
?ZYB6 value 2-428
?ZYB7 value 2-428
?ZYB8 value 2-428
?ZYB9 value 2-428
?ZYBF value 2-492
?ZYC0 value 2-434
?ZYC1 value 2-434
?ZYC2 value 2-434
?ZYCF value 2-495
?ZYD0 value 2-438
?ZYD1 value 2-438
?ZYD2 value 2-438
?ZYD3 value 2-438
?ZYD4 value 2-438
?ZYD5 value 2-438
?ZYD6 value 2-438
?ZYD7 value 2-438

?ZYD8 value 2-438
?ZYD9 value 2-438
?ZYDA value 2-438
?ZYDB value 2-438
?ZYE0 value 2-419
?ZYF0 value 2-423
?ZYF1 value 2-423
?ZYF2 value 2-423
?ZYG0 value 2-432
?ZYH0 value 2-436
?ZYI0 value 2-439
?ZYJ0 value 2-442
?ZYJ2 value 2-423
?ZYK0 value 2-471
?ZYK1 value 2-471
?ZYK2 value 2-471
?ZYK3 value 2-471
?ZYK4 value 2-471
?ZYK5 value 2-471
?ZYK6 value 2-471
?ZYK7 value 2-471
?ZYK8 value 2-471
?ZYK9 value 2-471
?ZYKA value 2-471
?ZYKB value 2-471
?ZYL0 value 2-421
?ZYM0 value 2-422
?ZYN0 value 2-424
?ZYO0 value 2-429
?ZYP0 value 2-431
?ZYPG value 2-440
?ZYQ0 value 2-459
?ZYR0 value 2-461
?ZYR1 value 2-461
?ZYR2 value 2-461
?ZYS0 value 2-465
?ZYS1 value 2-465
?ZYT0 value 2-469
?ZYU0 value 2-475
?ZYU1 value 2-475
?ZYU2 value 2-475
?ZYU3 value 2-475
?ZYU4 value 2-475
?ZYU5 value 2-475
?ZYU6 value 2-475
?ZYU7 value 2-475
?ZYV0 value 2-478
?ZYV1 value 2-478
?ZYV2 value 2-478
?ZYV3 value 2-478
?ZYV4 value 2-478
?ZYV5 value 2-478
?ZYV6 value 2-478
?ZYV7 value 2-478
?ZYV8 value 2-478
?ZYV9 value 2-478
?ZYVA value 2-478
?ZYVB value 2-478
?ZYVC value 2-478
?ZYVD value 2-478
?ZYVE value 2-478
?ZYW0 value 2-481
?ZYX0 value 2-487
?ZYY0 value 2-488
?ZYZ0 value 2-489
?ZZ00 value 2-435
?ZZ10 value 2-453
?ZZ20 value 2-455
?ZZ21 value 2-455
?ZZ22 value 2-455
?ZZ23 value 2-455
?ZZ24 value 2-455
?ZZ25 value 2-455
?ZZ26 value 2-455
?ZZ27 value 2-455
?ZZ28 value 2-455
?ZZ29 value 2-455
?ZZ2G value 2-455
?ZZ2H value 2-455
?ZZ2I value 2-455
?ZZ2J value 2-455
?ZZ30 value 2-458
?ZZ40 value 2-460
?ZZ50 value 2-466
?ZZ60 value 2-467
?ZZ70 value 2-486
?ZZ71 value 2-486
?ZZ72 value 2-486
?ZZ73 value 2-486
?ZZ74 value 2-486
?ZZ80 value 2-448
?ZZ81 value 2-448
?ZZ82 value 2-448
?ZZ83 value 2-448
?ZZ84 value 2-448
?ZZ85 value 2-448
?ZZ86 value 2-448
?ZZ87 value 2-448
?ZZ90 value 2-462
?ZZ9G value 2-418
?ZZ9H value 2-418
?ZZ9I value 2-418
?ZZAF value 2-491
?ZZAG value 2-491
?ZZAH value 2-491
?ZZB0 value 2-428
?ZZB1 value 2-428
?ZZB2 value 2-428
?ZZB3 value 2-428
?ZZB4 value 2-428
?ZZB5 value 2-428
?ZZB6 value 2-428

Licensed Material - Property of Data General Corporation

?ZZB7 value 2-428
?ZZB8 value 2-428
?ZZB9 value 2-428
?ZZBF value 2-492
?ZZC0 value 2-434
?ZZC1 value 2-434
?ZZC2 value 2-434
?ZZCF value 2-495
?ZZD0 value 2-438
?ZZD1 value 2-438
?ZZD2 value 2-438
?ZZD3 value 2-438
?ZZD4 value 2-438
?ZZD5 value 2-438
?ZZD6 value 2-438
?ZZD7 value 2-438
?ZZD8 value 2-438
?ZZD9 value 2-438
?ZZDA value 2-438
?ZZDB value 2-438
?ZZE0 value 2-419
?ZZF0 value 2-423
?ZZF1 value 2-423
?ZZF2 value 2-423
?ZZG0 value 2-432
?ZZH0 value 2-436
?ZZI0 value 2-439
?ZZJ0 value 2-442
?ZZK0 value 2-471
?ZZK1 value 2-471
?ZZK2 value 2-471
?ZZK3 value 2-471
?ZZK4 value 2-471
?ZZK5 value 2-471
?ZZK6 value 2-471
?ZZK7 value 2-471
?ZZK8 value 2-471
?ZZK9 value 2-471
?ZZKA value 2-471
?ZZKB value 2-471
?ZZL0 value 2-421
?ZZM0 value 2-422
?ZZN0 value 2-424
?ZZO0 value 2-429
?ZZP0 value 2-431
?ZZPG value 2-440
?ZZQ0 value 2-459
?ZZR0 value 2-461
?ZZR1 value 2-461
?ZZR2 value 2-461
?ZZS0 value 2-465
?ZZS1 value 2-465
?ZZT0 value 2-469
?ZZU0 value 2-475
?ZZU1 value 2-475
?ZZU2 value 2-475
?ZZU3 value 2-475
?ZZU4 value 2-475
?ZZU5 value 2-475
?ZZU6 value 2-475
?ZZU7 value 2-475
?ZZV0 value 2-478
?ZZV1 value 2-478
?ZZV2 value 2-478
?ZZV3 value 2-478
?ZZV4 value 2-478
?ZZV5 value 2-478
?ZZV6 value 2-478
?ZZV7 value 2-478
?ZZV8 value 2-478
?ZZV9 value 2-478
?ZZVA value 2-478
?ZZVB value 2-478
?ZZVC value 2-478
?ZZVD value 2-478
?ZZVE value 2-478
?ZZW0 value 2-481
?ZZX0 value 2-487
?ZZY0 value 2-488
?ZZZ0 value 2-489