

TOPS-20
Monitor Calls
Reference Manual

Electronically Distributed

This manual describes all the monitor calls that exist in the TOPS-20 operating system. For easy reference, the monitor call descriptions are arranged alphabetically and presented concisely. This manual supercedes the TOPS-20 Monitor Calls Reference Manual published in June, 1988. The part number for that manual, AA-FV52B-TM, is obsolete.

Change bars in the margins indicate material that has been added or changed since the previous printing of this manual.

Operating System:

TOPS-20 Version 7.0

| TOPS-20 Software Update Tape No. 04, November 1990

First Printing, September 1985

Revised, June 1988

| Revised, November 1990

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

| Copyright C 1985, 1988, 1990 Digital Equipment Corporation.

All Rights Reserved.

The following are trademarks of Digital Equipment Corporation:

| | | | |
|---------------|-----------|----------------|---------------|
| CI | DEctape | LA50 | SITGO-10 |
| DDCMP | DECUS | LN01 | TOPS-10 |
| DEC | DECwriter | LN03 | TOPS-20 |
| DECmail | DELNI | MASSEBUS | TOPS-20AN |
| DECnet | DELUA | PDP | UNIBUS |
| DECnet-VAX | HSC | PDP-11/24 | UETP |
| DECserver | HSC-50 | PrintServer | VAX |
| DECserver 100 | KA10 | PrintServer 40 | VAX/VMS |
| DECserver 200 | KI | Q-bus | VT50 |
| DECsystem-10 | KL10 | ReGIS | |
| DECSYSTEM-20 | KS10 | RSX | d i g i t a l |

CONTENTS

PREFACE

| | | |
|-----|---|----|
| 1 | REFERENCES | iv |
| 2 | OBSOLETE JSYSS | iv |
| 3 | CONVENTIONS USED IN THIS MANUAL | v |
| 3.1 | Number Bases | v |
| 3.2 | Abbreviations | v |
| 3.3 | Symbols | vi |
| 3.4 | Unimplemented Features | vi |

CHAPTER 1 INTRODUCTION

| | | |
|---------|--|------|
| 1.1 | CALLING CONVENTIONS | 1-2 |
| 1.2 | MONITOR CALL ARGUMENTS | 1-2 |
| 1.2.1 | Addresses | 1-3 |
| 1.2.2 | Page Numbers | 1-4 |
| 1.2.3 | Section Numbers | 1-4 |
| 1.2.4 | Byte Pointers | 1-4 |
| 1.2.5 | File Handles and File Designators | 1-6 |
| 1.2.6 | Source/Destination Designators | 1-6 |
| 1.2.6.1 | File Designator | 1-8 |
| 1.2.6.2 | Byte Pointers and ASCII Strings | 1-8 |
| 1.2.6.3 | Special Designators | 1-9 |
| 1.2.6.4 | Numeric Designators | 1-9 |
| 1.2.7 | Device Designator | 1-10 |
| 1.2.7.1 | Restrictions for Extended Addressing | 1-10 |
| 1.2.8 | Process Handles | 1-10 |
| 1.2.8.1 | Process/File Handle | 1-11 |
| 1.3 | SYSTEM DATE AND TIME | 1-11 |
| 1.4 | PROCESSING ERRORS | 1-12 |

CHAPTER 2 FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|---------|--|------|
| 2.1 | ACCOUNTING FUNCTIONS | 2-1 |
| 2.2 | REFERENCING FILES | 2-1 |
| 2.2.1 | File Specifications | 2-1 |
| 2.2.2 | Logical Names | 2-3 |
| 2.2.3 | File Handles | 2-3 |
| 2.2.4 | File References | 2-5 |
| 2.2.4.1 | Files and Devices | 2-6 |
| 2.2.5 | Sample Program | 2-6 |
| 2.2.6 | File Access | 2-9 |
| 2.2.7 | Directory Access | 2-10 |
| 2.2.8 | File Descriptor Block | 2-11 |
| 2.2.9 | Primary Input and Output Files | 2-22 |
| 2.2.10 | Methods of Data Transfer | 2-22 |
| 2.2.11 | File Byte Count | 2-22 |

| | | |
|----------|---|------|
| 2.2.12 | EOF Limit | 2-23 |
| 2.2.13 | Input/Output Errors | 2-23 |
| 2.2.13.1 | Testing for End-of-File | 2-24 |
| 2.3 | OBTAINING INFORMATION | 2-26 |
| 2.3.1 | Error Mnemonics and Message Strings | 2-26 |
| 2.3.2 | System Tables | 2-27 |
| 2.4 | COMMUNICATING WITH DEVICES | 2-34 |
| 2.4.1 | Physical Card Reader (PCDR:) | 2-36 |
| 2.4.2 | Spooled Card Reader (CDR:) | 2-37 |
| 2.4.3 | Physical Card Punch (PCDP:) | 2-37 |
| 2.4.4 | Spooled Card Punch (CDP:) | 2-38 |
| 2.4.5 | Physical Line Printer (PLPT:) | 2-38 |
| 2.4.5.1 | PLPT: Status Bits | 2-40 |
| 2.4.6 | Spooled Line Printer (LPT:) | 2-41 |
| 2.4.7 | Physical Magnetic Tape (MTA:) | 2-42 |
| 2.4.7.1 | Buffered I/O | 2-42 |
| 2.4.7.2 | Unbuffered I/O | 2-44 |
| 2.4.7.3 | Magnetic Tape Status | 2-44 |
| 2.4.7.4 | Reading a Tape in the Reverse Direction | 2-45 |
| 2.4.7.5 | Hardware Data Modes | 2-45 |
| 2.4.8 | Logical Magnetic Tape (MT:) | 2-48 |
| 2.4.9 | Terminal (TTY:) | 2-48 |
| 2.4.9.1 | JFN Mode Word | 2-49 |
| 2.4.9.2 | Control Character Output Control | 2-52 |
| 2.4.9.3 | Character Set | 2-52 |
| 2.4.9.4 | Terminal Characteristics Control | 2-55 |
| 2.4.9.5 | Terminal Linking | 2-58 |
| 2.4.9.6 | Terminal Advising | 2-58 |
| 2.4.10 | Transmission Control Protocol (TCP:) | 2-58 |
| 2.4.10.1 | GTJFN JSYS | 2-58 |
| 2.4.10.2 | OPENF JSYS | 2-59 |
| 2.4.10.3 | Other JSYSS | 2-60 |
| 2.5 | SOFTWARE DATA MODES | 2-61 |
| 2.6 | SOFTWARE INTERRUPT SYSTEM | 2-64 |
| 2.6.1 | Software Interrupt Channels | 2-64 |
| 2.6.2 | Software Interrupt Priority Levels | 2-66 |
| 2.6.3 | Software Interrupt Tables | 2-66 |
| 2.6.4 | Terminating Conditions | 2-67 |
| 2.6.5 | Panic Channels | 2-67 |
| 2.6.6 | Terminal Interrupts | 2-67 |
| 2.6.6.1 | Terminal Interrupt Modes | 2-70 |
| 2.6.7 | Dismissing an Interrupt | 2-70 |
| 2.7 | PROCESS CAPABILITIES | 2-72 |
| 2.7.1 | Assigned Capabilities | 2-72 |
| 2.7.2 | Access Control | 2-74 |
| 2.7.3 | Processes and Scheduling | 2-76 |
| 2.7.3.1 | Process Freezing | 2-76 |
| 2.7.3.2 | Execute-Only Files and Execute-Only Processes | 2-78 |
| 2.8 | SAVE FILES | 2-80 |
| 2.8.1 | Format for Nonsharable Save Files | 2-80 |
| 2.8.2 | Format of Sharable Save Files | 2-81 |
| 2.8.3 | Entry Vector | 2-84 |

| | | |
|---------|--|------|
| 2.8.4 | Program Data Vector | 2-85 |
| 2.9 | INPUT/OUTPUT CONVERSION | 2-86 |
| 2.9.1 | Floating Output Format Control | 2-86 |
| 2.9.1.1 | Free Format | 2-86 |
| 2.9.1.2 | General Format Control | 2-87 |
| 2.9.2 | Date And Time Conversion Monitor Calls | 2-89 |
| 2.10 | ARCHIVE/VIRTUAL DISK SYSTEM | 2-92 |
| 2.11 | PRIVILEGED MONITOR CALLS | 2-94 |

CHAPTER 3 TOPS-20 MONITOR CALLS

| | | |
|------|---------------------------|------|
| 3.1 | ACCES JSYS 552 | 3-1 |
| 3.2 | ADBRK JSYS 570 | 3-3 |
| 3.3 | AIC JSYS 131 | 3-7 |
| 3.4 | ALLOC JSYS 520 | 3-8 |
| 3.5 | ARCF JSYS 247 | 3-9 |
| 3.6 | ASND JSYS 70 | 3-13 |
| 3.7 | ASNIQ% JSYS 756 | 3-13 |
| 3.8 | ASNSQ JSYS 752 | 3-14 |
| 3.9 | ATACH JSYS 116 | 3-15 |
| 3.10 | ATI JSYS 137 | 3-17 |
| 3.11 | ATNVT JSYS 274 | 3-17 |
| 3.12 | BIN JSYS 50 | 3-18 |
| 3.13 | BKJFN JSYS 42 | 3-19 |
| 3.14 | BOOT JSYS 562 | 3-19 |
| 3.15 | BOUT JSYS 51 | 3-25 |
| 3.16 | CACCT JSYS 4 | 3-25 |
| 3.17 | CFIBF JSYS 100 | 3-26 |
| 3.18 | CFOBF JSYS 101 | 3-27 |
| 3.19 | CFORK JSYS 152 | 3-27 |
| 3.20 | CHFDB JSYS 64 | 3-29 |
| 3.21 | CHKAC JSYS 521 | 3-30 |
| 3.22 | CIS JSYS 141 | 3-31 |
| 3.23 | CLOSF JSYS 22 | 3-31 |
| 3.24 | CLZFF JSYS 34 | 3-33 |
| 3.25 | CNFIG% JSYS 627 | 3-34 |
| 3.26 | COMND JSYS 544 | 3-37 |
| 3.27 | CRDIR JSYS 240 | 3-61 |
| 3.28 | CRJOB JSYS 2 | 3-68 |
| 3.29 | CRLNM JSYS 502 | 3-75 |
| 3.30 | DEBRK JSYS 136 | 3-76 |
| 3.31 | DELDF JSYS 67 | 3-76 |
| 3.32 | DELF JSYS 26 | 3-78 |
| 3.33 | DELNF JSYS 317 | 3-79 |
| 3.34 | DEQ JSYS 514 | 3-80 |
| 3.35 | DEVST JSYS 121 | 3-82 |
| 3.36 | DFIN JSYS 234 | 3-82 |
| 3.37 | DFOUT JSYS 235 | 3-83 |
| 3.38 | DIAG JSYS 530 | 3-84 |
| 3.39 | DIBE JSYS 212 | 3-91 |
| 3.40 | DIC JSYS 133 | 3-92 |

| | | | |
|------|--------|----------------------|-------|
| 3.41 | DIR | JSYS 130 | 3-92 |
| 3.42 | DIRST | JSYS 41 | 3-93 |
| 3.43 | DISMS | JSYS 167 | 3-94 |
| 3.44 | DOB% | JSYS 635 | 3-94 |
| 3.45 | DOBE | JSYS 104 | 3-97 |
| 3.46 | DSKAS | JSYS 244 | 3-97 |
| 3.47 | DSKOP | JSYS 242 | 3-98 |
| 3.48 | DTACH | JSYS 115 | 3-101 |
| 3.49 | DTI | JSYS 140 | 3-101 |
| 3.50 | DUMPI | JSYS 65 | 3-102 |
| 3.51 | DUMPO | JSYS 66 | 3-103 |
| 3.52 | DVCHR | JSYS 117 | 3-105 |
| 3.53 | EIR | JSYS 126 | 3-106 |
| 3.54 | ENQ | JSYS 513 | 3-106 |
| 3.55 | ENQC | JSYS 515 | 3-113 |
| 3.56 | EPCAP | JSYS 151 | 3-117 |
| 3.57 | ERSTR | JSYS 11 | 3-118 |
| 3.58 | ESOUT | JSYS 313 | 3-118 |
| 3.59 | FFFFP | JSYS 31 | 3-119 |
| 3.60 | FFORK | JSYS 154 | 3-119 |
| 3.61 | FFUFP | JSYS 211 | 3-120 |
| 3.62 | FLIN | JSYS 232 | 3-120 |
| 3.63 | FLOUT | JSYS 233 | 3-121 |
| 3.64 | GACCT | JSYS 546 | 3-122 |
| 3.65 | GACTF | JSYS 37 | 3-122 |
| 3.66 | GCVEC | JSYS 300 | 3-123 |
| 3.67 | GDSKC | JSYS 214 | 3-123 |
| 3.68 | GDSTS | JSYS 145 | 3-124 |
| 3.69 | GDVEC | JSYS 542 | 3-125 |
| 3.70 | GET | JSYS 200 | 3-125 |
| 3.71 | GETAB | JSYS 10 | 3-128 |
| 3.72 | GETER | JSYS 12 | 3-129 |
| 3.73 | GETJI | JSYS 507 | 3-129 |
| 3.74 | GETNM | JSYS 177 | 3-131 |
| 3.75 | GETOK% | JSYS 574 | 3-132 |
| 3.76 | GEVEC | JSYS 205 | 3-142 |
| 3.77 | GFRKH | JSYS 164 | 3-142 |
| 3.78 | GFRKS | JSYS 166 | 3-143 |
| 3.79 | GFUST | JSYS 550 | 3-145 |
| 3.80 | GIVOK% | JSYS 576 | 3-146 |
| 3.81 | GJINF | JSYS 13 | 3-146 |
| 3.82 | GNJFN | JSYS 17 | 3-147 |
| 3.83 | GPJFN | JSYS 206 | 3-148 |
| 3.84 | GTAD | JSYS 227 | 3-148 |
| 3.85 | GTDAL | JSYS 305 | 3-149 |
| 3.86 | GTDIR | JSYS 241 | 3-149 |
| 3.87 | GTFDB | JSYS 63 | 3-151 |
| 3.88 | GTHST% | JSYS 273 | 3-151 |
| 3.89 | GTJFN | JSYS 20 (SHORT FORM) | 3-159 |
| 3.90 | GTJFN | JSYS 20 (LONG FORM) | 3-167 |
| 3.91 | GTRPI | JSYS 172 | 3-175 |
| 3.92 | GTRPW | JSYS 171 | 3-176 |

| | | | | |
|-------|--------|----------|-----------|-------|
| 3.93 | GTSTS | JSYS 24 | | 3-177 |
| 3.94 | GTTYP | JSYS 303 | | 3-178 |
| 3.95 | HALTF | JSYS 170 | | 3-178 |
| 3.96 | HFORK | JSYS 162 | | 3-179 |
| 3.97 | HPTIM | JSYS 501 | | 3-179 |
| 3.98 | HSYS | JSYS 307 | | 3-180 |
| 3.99 | IDCNV | JSYS 223 | | 3-181 |
| 3.100 | IDTIM | JSYS 221 | | 3-182 |
| 3.101 | IDTNC | JSYS 231 | | 3-184 |
| 3.102 | IIC | JSYS 132 | | 3-186 |
| 3.103 | INFO% | JSYS 633 | | 3-186 |
| 3.104 | INLNM | JSYS 503 | | 3-195 |
| 3.105 | IPOPR% | JSYS 760 | | 3-196 |
| 3.106 | JFNS | JSYS 30 | | 3-197 |
| 3.107 | KFORK | JSYS 153 | | 3-200 |
| 3.108 | LATOP% | JSYS 631 | | 3-200 |
| 3.109 | LGOUT | JSYS 3 | | 3-215 |
| 3.110 | LLMOP% | JSYS 624 | | 3-216 |
| 3.111 | LN MST | JSYS 504 | | 3-224 |
| 3.112 | LOGIN | JSYS 1 | | 3-225 |
| 3.113 | LPINI | JSYS 547 | | 3-226 |
| 3.114 | MDDT% | JSYS 777 | | 3-227 |
| 3.115 | METER% | JSYS 766 | | 3-227 |
| 3.116 | MRECV | JSYS 511 | | 3-229 |
| 3.117 | MSEND | JSYS 510 | | 3-231 |
| 3.118 | MSFRK | JSYS 312 | | 3-235 |
| 3.119 | MSTR | JSYS 555 | | 3-236 |
| 3.120 | MTALN | JSYS 774 | | 3-257 |
| 3.121 | MTOPR | JSYS 77 | | 3-257 |
| 3.122 | MTU% | JSYS 600 | | 3-293 |
| 3.123 | MUTIL | JSYS 512 | | 3-295 |
| 3.124 | NI% | JSYS 630 | | 3-302 |
| 3.125 | NIN | JSYS 225 | | 3-320 |
| 3.126 | NODE | JSYS 567 | | 3-321 |
| 3.127 | NOUT | JSYS 224 | | 3-329 |
| 3.128 | NTINF% | JSYS 632 | | 3-330 |
| 3.129 | NTMAN% | JSYS 604 | | 3-332 |
| 3.130 | ODCNV | JSYS 222 | | 3-334 |
| 3.131 | ODTIM | JSYS 220 | | 3-336 |
| 3.132 | ODTNC | JSYS 230 | | 3-338 |
| 3.133 | OPENF | JSYS 21 | | 3-339 |
| 3.134 | PBIN | JSYS 73 | | 3-344 |
| 3.135 | PBOUT | JSYS 74 | | 3-345 |
| 3.136 | PDVOP% | JSYS 603 | | 3-345 |
| 3.137 | PEEK | JSYS 311 | | 3-348 |
| 3.138 | PLOCK | JSYS 561 | | 3-349 |
| 3.139 | PMAP | JSYS 56 | | 3-350 |
| 3.140 | PMCTL | JSYS 560 | | 3-355 |
| 3.141 | PPNST | JSYS 557 | | 3-358 |
| 3.142 | PRARG | JSYS 545 | | 3-359 |
| 3.143 | PSOUT | JSYS 76 | | 3-361 |
| 3.144 | QUEUE% | JSYS 615 | | 3-361 |

| | | | | |
|-------|--------|----------|-----------|-------|
| 3.145 | RCDIR | JSYS 553 | | 3-368 |
| 3.146 | RCM | JSYS 134 | | 3-372 |
| 3.147 | RCUSR | JSYS 554 | | 3-372 |
| 3.148 | RCVIM | JSYS 751 | | 3-374 |
| 3.149 | RCVIN% | JSYS 755 | | 3-375 |
| 3.150 | RCVOK% | JSYS 575 | | 3-376 |
| 3.151 | RDTTY | JSYS 523 | | 3-377 |
| 3.152 | RELD | JSYS 71 | | 3-380 |
| 3.153 | RELIQ% | JSYS 757 | | 3-380 |
| 3.154 | RELSQ | JSYS 753 | | 3-381 |
| 3.155 | RESET | JSYS 147 | | 3-381 |
| 3.156 | RFACS | JSYS 161 | | 3-382 |
| 3.157 | RFBSZ | JSYS 45 | | 3-383 |
| 3.158 | RFCOC | JSYS 112 | | 3-383 |
| 3.159 | RFMOD | JSYS 107 | | 3-384 |
| 3.160 | RFORK | JSYS 155 | | 3-385 |
| 3.161 | RFPOS | JSYS 111 | | 3-385 |
| 3.162 | RFPTR | JSYS 43 | | 3-386 |
| 3.163 | RFRKH | JSYS 165 | | 3-386 |
| 3.164 | RFSTS | JSYS 156 | | 3-387 |
| 3.165 | RFTAD | JSYS 533 | | 3-390 |
| 3.166 | RIN | JSYS 54 | | 3-391 |
| 3.167 | RIR | JSYS 144 | | 3-392 |
| 3.168 | RIRCM | JSYS 143 | | 3-393 |
| 3.169 | RLJFN | JSYS 23 | | 3-393 |
| 3.170 | RMAP | JSYS 61 | | 3-394 |
| 3.171 | RNAMF | JSYS 35 | | 3-394 |
| 3.172 | ROUT | JSYS 55 | | 3-396 |
| 3.173 | RPACS | JSYS 57 | | 3-397 |
| 3.174 | RPCAP | JSYS 150 | | 3-398 |
| 3.175 | RSCAN | JSYS 500 | | 3-398 |
| 3.176 | RSMAP% | JSYS 610 | | 3-400 |
| 3.177 | RTFRK | JSYS 322 | | 3-401 |
| 3.178 | RTIW | JSYS 173 | | 3-402 |
| 3.179 | RUNTM | JSYS 15 | | 3-402 |
| 3.180 | RWM | JSYS 135 | | 3-403 |
| 3.181 | RWSET | JSYS 176 | | 3-404 |
| 3.182 | SACTF | JSYS 62 | | 3-404 |
| 3.183 | SAVE | JSYS 202 | | 3-405 |
| 3.184 | SCS% | JSYS 622 | | 3-406 |
| 3.185 | SCTTY | JSYS 324 | | 3-423 |
| 3.186 | SCVEC | JSYS 301 | | 3-424 |
| 3.187 | SDSTS | JSYS 146 | | 3-426 |
| 3.188 | SDVEC | JSYS 543 | | 3-426 |
| 3.189 | SETER | JSYS 336 | | 3-427 |
| 3.190 | SETJB | JSYS 541 | | 3-428 |
| 3.191 | SETNM | JSYS 210 | | 3-431 |
| 3.192 | SETSN | JSYS 506 | | 3-431 |
| 3.193 | SEVEC | JSYS 204 | | 3-431 |
| 3.194 | SFACS | JSYS 160 | | 3-432 |
| 3.195 | SFBSZ | JSYS 46 | | 3-433 |
| 3.196 | SFCOC | JSYS 113 | | 3-433 |

| | | | | |
|-------|--------|----------|-----------|-------|
| 3.197 | SFMOD | JSYS 110 | | 3-434 |
| 3.198 | SFORK | JSYS 157 | | 3-435 |
| 3.199 | SFPOS | JSYS 526 | | 3-436 |
| 3.200 | SFPTR | JSYS 27 | | 3-436 |
| 3.201 | SFRKV | JSYS 201 | | 3-438 |
| 3.202 | SFTAD | JSYS 534 | | 3-439 |
| 3.203 | SFUST | JSYS 551 | | 3-441 |
| 3.204 | SIBE | JSYS 102 | | 3-442 |
| 3.205 | SIN | JSYS 52 | | 3-443 |
| 3.206 | SINR | JSYS 531 | | 3-444 |
| 3.207 | SIR | JSYS 125 | | 3-446 |
| 3.208 | SIRCM | JSYS 142 | | 3-447 |
| 3.209 | SIZEF | JSYS 36 | | 3-447 |
| 3.210 | SJPRI | JSYS 245 | | 3-448 |
| 3.211 | SKED% | JSYS 577 | | 3-449 |
| 3.212 | SKPIR | JSYS 127 | | 3-455 |
| 3.213 | SMAP% | JSYS 767 | | 3-455 |
| 3.214 | SMON | JSYS 6 | | 3-459 |
| 3.215 | SNDIM | JSYS 750 | | 3-464 |
| 3.216 | SNDIN% | JSYS 754 | | 3-465 |
| 3.217 | SNOOP | JSYS 516 | | 3-466 |
| 3.218 | SOBE | JSYS 103 | | 3-469 |
| 3.219 | SOBF | JSYS 175 | | 3-470 |
| 3.220 | SOUT | JSYS 53 | | 3-470 |
| 3.221 | SOUTR | JSYS 532 | | 3-472 |
| 3.222 | SPACS | JSYS 60 | | 3-473 |
| 3.223 | SPJFN | JSYS 207 | | 3-474 |
| 3.224 | SPLFK | JSYS 314 | | 3-475 |
| 3.225 | SPOOL | JSYS 517 | | 3-477 |
| 3.226 | SPRIW | JSYS 243 | | 3-479 |
| 3.227 | SSAVE | JSYS 203 | | 3-480 |
| 3.228 | STAD | JSYS 226 | | 3-482 |
| 3.229 | STCMP | JSYS 540 | | 3-482 |
| 3.230 | STDEV | JSYS 120 | | 3-483 |
| 3.231 | STI | JSYS 114 | | 3-484 |
| 3.232 | STIW | JSYS 174 | | 3-485 |
| 3.233 | STO | JSYS 246 | | 3-486 |
| 3.234 | STPAR | JSYS 217 | | 3-487 |
| 3.235 | STPPN | JSYS 556 | | 3-488 |
| 3.236 | STSTS | JSYS 25 | | 3-489 |
| 3.237 | STTYP | JSYS 302 | | 3-490 |
| 3.238 | SWJFN | JSYS 47 | | 3-490 |
| 3.239 | SWTRP% | JSYS 573 | | 3-491 |
| 3.240 | SYERR | JSYS 527 | | 3-492 |
| 3.241 | SYSGT | JSYS 16 | | 3-493 |
| 3.242 | TBADD | JSYS 536 | | 3-493 |
| 3.243 | TBDEL | JSYS 535 | | 3-494 |
| 3.244 | TBLUK | JSYS 537 | | 3-495 |
| 3.245 | TCOPR% | JSYS 761 | | 3-497 |
| 3.246 | TEXTI | JSYS 524 | | 3-499 |
| 3.247 | TFORK | JSYS 321 | | 3-504 |
| 3.248 | THIBR | JSYS 770 | | 3-507 |

| | | | |
|-------|--------|----------|-------|
| 3.249 | TIME | JSYS 14 | 3-507 |
| 3.250 | TIMER | JSYS 522 | 3-507 |
| 3.251 | TLINK | JSYS 216 | 3-509 |
| 3.252 | TMON | JSYS 7 | 3-511 |
| 3.253 | TTMSG | JSYS 775 | 3-514 |
| 3.254 | TWAKE | JSYS 771 | 3-515 |
| 3.255 | UFPGS | JSYS 525 | 3-516 |
| 3.256 | USAGE | JSYS 564 | 3-516 |
| 3.257 | USRIO | JSYS 310 | 3-520 |
| 3.258 | UTEST | JSYS 563 | 3-520 |
| 3.259 | UTFRK | JSYS 323 | 3-521 |
| 3.260 | VACCT | JSYS 566 | 3-523 |
| 3.261 | WAIT | JSYS 306 | 3-523 |
| 3.262 | WFORK | JSYS 163 | 3-524 |
| 3.263 | WILD% | JSYS 565 | 3-524 |
| 3.264 | WSMGR% | JSYS 623 | 3-526 |
| 3.265 | XGSEV% | JSYS 614 | 3-527 |
| 3.266 | XGTPW% | JSYS 612 | 3-528 |
| 3.267 | XGVEC% | JSYS 606 | 3-529 |
| 3.268 | XPEEK% | JSYS 626 | 3-529 |
| 3.269 | XRIR% | JSYS 601 | 3-531 |
| 3.270 | XRMAP% | JSYS 611 | 3-531 |
| 3.271 | XSFRK% | JSYS 605 | 3-533 |
| 3.272 | XSIR% | JSYS 602 | 3-533 |
| 3.273 | XSSEV% | JSYS 613 | 3-534 |
| 3.274 | XSVEC% | JSYS 607 | 3-535 |

APPENDIX A ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

APPENDIX B TOPS-20 ERROR CODES AND MNEMONICS

INDEX

TABLES

| | | |
|------|--|------|
| 1-1 | P-Field Values for One-word Global Byte Pointers | 1-5 |
| 1-2 | Source/Destination Designators | 1-7 |
| 2-1 | File Descriptor Block (FDB) | 2-12 |
| 2-2 | System Tables | 2-27 |
| 2-3 | Device Types | 2-35 |
| 2-4 | PCDR: Status Bits | 2-36 |
| 2-5 | PCDP: Status Bits | 2-37 |
| 2-6 | PLPT: Control Characters | 2-39 |
| 2-7 | PLPT: Status Bits | 2-40 |
| 2-8 | MTA: Status Bits | 2-42 |
| 2-9 | JFN Mode Word | 2-49 |
| 2-10 | WakeUp Classes/CCOC Word Bits | 2-53 |

| | | |
|------|---|------|
| 2-11 | Terminal Characteristics | 2-55 |
| 2-12 | Software Interrupt Channels | 2-65 |
| 2-13 | Terminal Interrupt Codes | 2-68 |
| 2-14 | Process/Job Capabilities | 2-72 |
| 2-15 | Floating-Point Format Control | 2-87 |
| 2-16 | Time Zones | 2-91 |
| A-1 | ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC | A-1 |
| A-2 | EBCDIC Collating Sequence and Conversion to ASCII | A-4 |

PREFACE

This manual is written for the assembly language programmer who is already familiar with TOPS-20 monitor calls. For an introductory discussion of some basic monitor calls, refer to the TOPS-20 Monitor Calls User's Guide.

Chapter 1 introduces the conventions to follow when using monitor calls, and describes the types of arguments used with the monitor calls. Chapter 2 presents the calls related to particular functions and tasks, such as using the software interrupt system. Chapter 3 contains, in alphabetical order, descriptions of all the monitor calls.

Appendix A contains the EBCDIC, ASCII, and SIXBIT collating sequences, and conversions between these three character set representations. Appendix B contains a numeric list of error codes with their corresponding mnemonic, and an alphabetic list of mnemonics with their corresponding code and text string.

1 REFERENCES

The following publications are either referenced in this manual or are recommended as supplements to this manual:

| Referenced as | Title |
|----------------------------|---|
| Monitor Calls User's Guide | <u>TOPS-20 Monitor Calls User's Guide</u> |
| System Administrator | <u>TOPS-20 System Manager's Guide</u> |
| TCP/IP Handbook | <u>Internet Protocol Transition Workbook</u> |
| | Available from: |
| | Network Information Center SRI International Menlo Park, California 94025 |
| DECnet Manual | <u>DECnet-20 User's Guide</u> |
| Assembler Manual | <u>MACRO Assembler Reference Manual</u> |
| Link Manual | <u>TOPS-20 LINK Reference Manual</u> |
| Hardware Reference Manual | <u>DECsystem-10/DECSYSTEM-20 Processor Reference Manual</u> |
| Commands Reference Manual | <u>TOPS-20 Commands Reference Manual</u> |
| RMS Manual | <u>TOPS-20 RMS User's Guide</u> |
| SPEAR Manual | <u>TOPS-10/TOPS-20 SPEAR Manual</u> |
| TOPS-20 User's Guide | <u>TOPS-20 User's Guide</u> |
| Installation Guide | <u>TOPS-20 KL Model B Installation Guide</u> |
| Network Management Spec | <u>Network Management Architecture Specification</u> |

2 OBSOLETE JSYSS

The following JSYSSs are obsolete as of version V6.1 of TOPS-20:

CVHST

CVSKT

FLHST

GTNCP

3 CONVENTIONS USED IN THIS MANUAL

3.1 Number Bases

Except where otherwise noted, numbers used in this manual, including those in the definition of a monitor call description, are octal. When indicated, bits in words are numbered in decimal with the leftmost bit of the word labeled B0 and the rightmost bit of the word labeled B35.

3.2 Abbreviations

The following abbreviations are used in this manual:

| | |
|-------------|--|
| B0, B1, ... | Bit 0, bit 1, ... of the computer word |
| nBm | Field whose rightmost bit is m and whose value is n (5B2, for example). |
| LH | Left Half (B0-B17 of the word) |
| RH | Right Half (B18-B35 of the word) |
| JFN | Job File Number |
| PSB | Process Storage Block (a table containing all monitor data for the process) |
| JSB | Job Storage Block (a table containing all monitor data relevant to the job) |
| CCOC words | Control Character Output Control words (2 words containing 36 2-bit bytes that determine the way in which control characters are output. Refer to Section 2.4.9.2.) |
| FDB | File Descriptor Block (a table in a file that contains information about the file). Refer to Section 2.2.8. |
| TCP/IP | Transmission Control Protocol/Internet Protocol |

3.3 Symbols

The symbols used in this manual, including the names of the monitor calls, are defined in the system file MONSYM.MAC. A program that uses a monitor call or other symbol must include the statement

```
SEARCH MONSYM
```

before the first occurrence of a symbol. Failure to include this statement causes errors in the compilation of the program.

The system file MACSYM.MAC contains a number of useful macros for the assembly language programmer. To use MACSYM macros, the user's program must contain the statements

```
SEARCH MACSYM
.REQUIRE SYS:MACREL ;include support routines
```

at the beginning of the program. Since most bits defined for use with the monitor have symbolic names, macros enable the programmer to use these bits without knowledge of their exact position. Refer to the Monitor Calls User's Guide for more information on MACSYM macros.

3.4 Unimplemented Features

The MONSYM file contains symbol names for several monitor calls and bit positions that are not described in this manual. These features are not implemented in TOPS-20.

If an unimplemented monitor call is used in a user program, it causes an illegal instruction interrupt unless followed by an ERJMP or ERCAL symbol. In this case, the ERJMP will be executed. It is recommended that unimplemented or undefined bit positions be zero to allow for future expansion.

CHAPTER 1

INTRODUCTION

The TOPS-20 Monitor Calls Reference Manual describes every monitor call in the TOPS-20 system. Monitor calls for TCP/IP systems and DECnet systems are also described.

TOPS-20 monitor calls invoke the TOPS-20 monitor by means of the JSYS instruction (op code 104). The UUO-type monitor calls (op codes 40-77) invoke the TOPS-10 compatibility package, which simulates the action of these UUO's in the TOPS-10 monitor. Programs written for TOPS-20 should use TOPS-20 monitor calls, not UUO's.

For easy reference, monitor call descriptions in Chapter 3 are arranged alphabetically and presented concisely. This concise format begins with the monitor call name and numeric definition, followed by a brief description of the monitor call function. The calling sequence for the monitor call is next, indicated by statements in the format

ACCEPTS IN ACn: description

where n is an accumulator number. Following the list of accumulators and descriptions of their contents are statements of the form

RETURNS +1: condition
 +2: condition

These statements define where control returns, and under what conditions, after execution of the monitor call. The statement RETURNS+1: means that control returns to the memory location immediately following the calling location. The statement RETURNS+2: means that control returns to the second memory location after the calling location.

Next, there is an optional description of the action taken by the monitor call.

INTRODUCTION

1.1 CALLING CONVENTIONS

Arguments for the monitor call are placed in accumulators (ACs), then the monitor call is executed. The first argument is in AC1, the second in AC2, and so forth.

Many calls also require an argument block. This is a group of contiguous words of memory that contain additional arguments. If an argument block is required, an AC must contain a pointer to the argument block. See the description of the GTJFN% monitor call for an example of the use of argument blocks.

In addition, arguments in an argument block can point to other argument blocks. These other argument blocks can, in turn, contain other groups of arguments. For an example of this way of passing many arguments to a monitor call, see the description of the GTJFN call in Chapter 3. (There are several exceptions to this convention; refer to the individual descriptions in Chapter 3.)

Data returned by the execution of a monitor call is often returned in the ACs. If a call returns more data than can be held in four ACs, it returns the data to a data block. A pointer to the data block must be passed as an argument to the monitor call. Such a pointer can be passed in either an AC or an argument block.

When using a monitor call in a program, end the name of the call with a percent (%) character. This convention helps avoid conflicts between monitor call names and symbols defined by your programs. In addition, this convention is required by monitor calls defined in TOPS-20 Version 4.0 or later. Although older calls do not require a percent character at the end of their names, they will accept one.

1.2 MONITOR CALL ARGUMENTS

A monitor call argument can be one of the following:

- o a word of data
- o the memory address word that contains data
- o a page number
- o a section number
- o a byte pointer
- o a file handle
- o a source (or destination) designator that defines where to obtain (or send) data

INTRODUCTION

- o a process handle
- o a file/process handle

The following sections describe these arguments.

1.2.1 Addresses

On a DECSYSTEM-20, addresses can be one of two types: an 18-bit address, or a 30-bit address. TOPS-20 supports 30-bit addressing, but currently allows access to an address space of 32 (decimal) sections, each of which contains 256K words. Therefore, although a global address is said to be a 30-bit address, only the rightmost 23 bits are meaningful: five bits of section number and 18 bits of in-section address.

An 18-bit address is called a local (section-relative) address. With such an address you can specify any word in a 256K-word section of memory, but you cannot also specify a section number. With a 30-bit, or global, address you can reference any word of any section of memory. (Refer to the Hardware Reference Manual for a description of global addresses.)

TOPS-20 allows you to use 18-bit or 30-bit addresses. Some monitor calls require one kind, some the other; some calls accept either kind.

Some monitor calls use only 18 bits to hold an address. These calls interpret 18-bit addresses as locations in the current section, the same section as that of the code being executed (the same section as the user PC.) To form an unambiguous global address, these calls add the section number of the PC to the section-relative address.

Monitor calls that use an entire word for an address can accept either 18-bit or 30-bit addresses. If the address is 30 bits (the section number is not 0), it is a global address.

If the address is 18 bits (the section number is 0), the monitor call acts in one of two ways. If the call existed in Release 4 or earlier, it interprets the address as a section-relative address, as stated above. But if the call is one of the extended-addressing calls (if the call starts with an X), the call interprets the zero in the section-number field as indicating section 0.

It is sometimes desirable to specify addresses in section 0 when executing a JSYS from a nonzero section. The bit PM%EPN for PMAP%, and FH%EPN for JSYSs that accept fork handles, prevent the current section (the section in which the program is running) from being the target section for the monitor call's arguments.

INTRODUCTION

1.2.2 Page Numbers

A TOPS-20 page number can be 9 bits or 18 bits long. A page number can refer to either a page of memory, or a page of a disk file.

The 9-bit number is called a section-relative page number. Such a page number can specify any page within a 256K-word section of memory, or any page within a 256K section of a file. (A file section is a unit of 512 pages within a file. The first page of each such section has a page number that is an integer multiple of 512.)

The left half of a section-relative (18-bit) address can be considered to be a section-relative page number. If a monitor call uses only 9 bits of a word to hold a page number, the monitor considers that page to be within the current section.

Most monitor calls that require page numbers as arguments use at least half of a word to contain the page number. Such calls allow you to specify an 18-bit, or global, page number. A global page number refers to both a section of memory and a page within that section. Page 23200, for example, is page 200 in section 23.

1.2.3 Section Numbers

A section number is five bits long. In a global address, a section number occupies bits 13 through 17. Because TOPS-20 supports 40 (octal) sections of memory, using section numbers larger than 37 causes an error.

1.2.4 Byte Pointers

Monitor calls accept two kinds of byte pointers as arguments: one-word local byte pointers, and one-word global byte pointers. One-word local byte pointers work in all sections, but one-word global byte pointers cannot be used in section 0.

The Hardware Reference Manual describes one-word local byte pointers in detail. The following paragraphs discuss one-word global byte pointers.

Any monitor calls that accept source/destination designators (See Section 1.2.6.) also accept byte pointers, and the bytes can be from 1 to 36 bits long. SIN and SOUT are examples of such monitor calls.

If a call cannot accept a source/destination designator, however, that call only accepts byte pointers that point to 7-bit bytes. Examples of such calls are CACCT and PSOUT. Note, however, that for historical reasons some monitor calls accept one-word global byte pointers that point to bytes of other lengths.

INTRODUCTION

TOPS-20 monitor calls do not accept the two-word local byte pointers or the two-word global byte pointers described in the Hardware Reference Manual.

Local byte pointers can only point to a byte in the current section. This is because they use 18 bits to hold the address of the byte. You can use indexing with local byte pointers, however, to point to a byte in another section of memory.

If, for example, AC5 contains a 30-bit address, the following instruction generates an indexed local byte pointer in AC2. The pointer points to a byte in another section, the section of the address in AC5.

```
MOVE 2,[POINT 7,0(5)]
```

Use of indirect addressing with local byte pointers is discouraged.

Global byte pointers use 30 bits to hold the address of the byte, thus they can point to a byte in any section of memory. One-word global byte pointers have the following format:

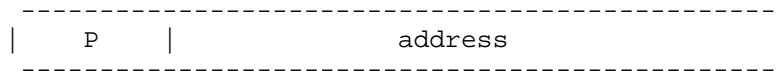


Table 1-1 shows how the KL-10 processor interprets the P field.

Table 1-1: P-Field Values for One-word Global Byte Pointers

| P (octal) | Byte Size | Position of the Right-Most Bit (count, in octal, of the number of bits to the right of the current pointer position) |
|--------------|-----------------------|---|
| Less than 45 | a local byte pointer. | |
| 45 | 6 | 44 |
| 46 | 6 | 36 |
| 47 | 6 | 30 |
| 50 | 6 | 22 |
| 51 | 6 | 14 |
| 52 | 6 | 6 |
| 53 | 6 | 0 |
| 54 | 8 | 44 |

INTRODUCTION

| | | |
|----|---|----|
| 55 | 8 | 34 |
| 56 | 8 | 24 |
| 57 | 8 | 14 |
| 60 | 8 | 4 |
| 61 | 7 | 44 |
| 62 | 7 | 35 |
| 63 | 7 | 26 |
| 64 | 7 | 17 |
| 65 | 7 | 10 |
| 66 | 7 | 1 |
| 67 | 9 | 44 |
| 70 | 9 | 33 |
| 71 | 9 | 22 |
| 72 | 9 | 11 |
| 73 | 9 | 0 |
| 74 | 18 | 44 |
| 75 | 18 | 22 |
| 76 | 18 | 0 |
| 77 | unused (causes an illegal instruction trap) | |

You cannot use indexing or indirect addressing with one-word global byte pointers.

1.2.5 File Handles and File Designators

A file handle is also known as a job file number, or JFN. It is an 18-bit number that, within the context of a job, uniquely identifies a file.

An indexable file handle, or full-word JFN, has a JFN in the right half and flags in the left half. This file handle is useful for handling several files in sequence. See Section 2.2.3 for a more complete discussion of file handles.

1.2.6 Source/Destination Designators

Some monitor calls act upon bytes or strings of bytes, or transfer bytes from one place to another. Such calls often use source/destination designators to identify where the bytes are sent or obtained.

A source/destination designator is a 36-bit quantity that can have the

INTRODUCTION

formats given in Table 1-2. The paragraphs following the table describe each designator. Note that byte pointers are also source/destination designators.

Table 1-2: Source/Destination Designators

| Symbol | Left Half | Right Half | Meaning |
|--------|-----------|------------|--|
| (none) | 0 | JFN | a job file number. The JFN is the job's handle on a file, and is assigned with the GTJFN monitor call. (Refer to Section 2.2.3.) |
| .PRIIN | 0 | 100 | primary input designator |
| .PRIOU | 0 | 101 | primary output designator |
| .NULIO | 0 | 377777 | null designator |
| .TTDES | 0 | 4xxxxx | universal terminal designator |
| .SIGIO | 0 | 677777 | signal JFN. When a fork's I/O designator is .SIGIO, then any attempt to perform I/O to that JFN will freeze the fork and cause a channel 19 (fork termination) interrupt to be sent to that fork's superior. |
| .CTTRM | 0 | 777777 | the job's controlling terminal |
| .DVDES | 6xxxxx | xxxxxx | universal device designator (for use only in section 0) |
| | 777777 | address | implicit byte pointer. TOPS-20 changes left half to 440700. (Refer to Sections 1.2.4 and 1.2.6.2.) .b.i-35 777777 777777 universal default |
| | 5xxxxx | xxxxxx | numeric value |

Note: The designators .PRIIN and .PRIOU are legal wherever a JFN is expected. You cannot assign them as JFN's, however. GTJFN and GNJFN never assign 100 or 101.

INTRODUCTION

The most commonly used source/destination designators are:

1. A JFN, identifying a particular file. Before a JFN can be used, it must be obtained by means of the GTJFN monitor call. (See Section 2.2.3.)
2. The primary input and output designators. (Refer to Section 2.2.9.) These designators are the ones recommended for use in referring to the job's controlling terminal because they can be changed to cause terminal input and/or output to be taken from and/or sent to a file. The controlling terminal designator .CTTRM (0,-1) cannot be redirected in this way, and its use is not recommended in normal situations.
3. A byte pointer to the beginning of the string being read or written.

1.2.6.1 File Designator - A file designator indicates that I/O to be done by the monitor call is to be done as though to a terminal. A file designator can be any of the following: .PRIIN, .PRIOU, .NULIO, .TTDES, .CTTRM, or .DVDES.

1.2.6.2 Byte Pointers and ASCII Strings - Many monitor calls deal specifically with ASCII strings. The following conventions apply to such strings.

1. A file designator can be used if the file is in 7-bit ASCII format. This is the usual format for text files.
2. One of the following is used to designate a string in the caller's address space:
 - a. -1,,ADR to designate a 7-bit ASCII string beginning in the leftmost byte of ADR. This is for convenience, making HRROI 1,ADR functionally equivalent to MOVE 1,[POINT 7,ADR].
 - b. A byte pointer with a byte size of 7 bits. If the byte size is not 7 bits, the results might be incorrect. This is because monitor calls use the ILDB and IDPB instructions to reference byte strings, and do no additional checking to see that the data is in the correct format. Note, however, that for historical reasons some monitor calls accept byte pointers with byte sizes larger or smaller than 7 bits.

INTRODUCTION

NOTE

Unless otherwise noted, the term "byte pointer" is used in this manual to indicate an ILDB/IDPB byte pointer that points to an ASCIZ string. The following example generates such a byte pointer:

```
POINT 7,[ASCIZ/character string/]
```

The term "pointer" is usually used to refer to an address, except in discussions that must make repeated references to the term "byte pointer". In the latter case, some of the occurrences of "byte pointer" will be shortened to "pointer" to avoid monotonous repetition. In these cases, however, it will be clear from the context that "pointer" implies "byte pointer".

Normally, monitor calls assume that ASCII strings are terminated with a byte containing zeroes (an ASCIZ string). A few calls terminate on other ASCII characters because of context (the NIN call, for example), and some optionally accept an explicit byte count or allow you to determine the terminating byte. These latter calls (SIN and SOUT calls, for example) are generally those that can handle non-ASCII strings and byte sizes other than 7 bits.

After a monitor call is used to read a string, the source byte pointer argument is updated such that an ILDB would read the character following the terminating character; an LDB would reread the terminating character.

After a monitor call is used to write a string, the destination byte pointer argument is updated to point to the character following the last nonnull character written. If there is room, a null byte is appended to the string, but the byte pointer returned is such that an IDPB will overwrite the null.

1.2.6.3 Special Designators - The universal default designator of -1 is used to indicate the current designator, such as the current job or the connected directory. For example, the GETJI monitor call accepts an argument of -1 as the designator for the current job.

1.2.6.4 Numeric Designators - The designator 5xxxxx xxxxxx (where a numeric value is in bits 3-35) is used to supply a numeric designator as an argument to a call. Numeric designators are used to identify account numbers, directory numbers, user numbers, and the like. The DIRST monitor call, for example, accepts a user number as 5B2+33-bit number.

INTRODUCTION

1.2.7 Device Designator

Many monitor calls dealing with devices (refer to Section 2.4) take a device designator as an argument. A device designator can be either

```
LH: .DVDES(600000)+device type number
RH: unit number for devices that have units, arbitrary code for
    structures, or -1 for nonstructure devices that do not have
    units
```

or

```
LH: 0
RH: .TTDES(400000)+ terminal number, or .CTTRM(777777) for
    controlling terminal
```

Thus, terminals can be represented in two ways; the second way is provided for compatibility with the source/destination designator.

Because designators for structures contain an arbitrary code, these designators must always be obtained from the monitor (by means of the STDEV call) and cannot be created by the program.

Section 2.4 describes the various devices and their type numbers.

1.2.7.1 Restrictions for Extended Addressing - A restriction on arguments passed to monitor calls executed in sections other than section 0 concerns universal device designators and numeric designators, which have the format 5xxxxxx,xxxxxx or 6xxxxxx,xxxxxx (.DVDES). These designators are only legal in section 0. This is because of the existence of one-word global byte pointers, which can have the same format.

Thus, monitor calls that accept either this type of designator or a byte pointer when called from section 0 do not accept these designators in any other section. Other device designators, such as .TTDES (0,,4xxxxxx), can be used in any section. Conversely, these monitor calls that can accept either device/numeric designators or byte pointers do not accept one-word global byte pointers in section 0.

1.2.8 Process Handles

Several monitor calls accept an 18-bit argument called a process handle. The following fork handles are defined within the context of a job.

INTRODUCTION

| Value | Symbol | Meaning |
|----------|--------|---|
| 400000 | .FHSLF | Current process |
| 400000+n | - | Process n, relative to the current process |
| 200000 | FH%EPN | Extended page number. When used in conjunction with the above two forms, this bit indicates that addresses and/or page numbers are interpreted as absolute, NOT relative to the PC section of the program executing the JSYS. This bit has no meaning for programs that do not use extended addressing. |
| -1 | .FHSUP | Superior process |
| -2 | .FHTOP | Top-level process |
| -3 | .FHSAI | Current process and all of its inferiors |
| -4 | .FHINF | All of the current process's inferiors |
| -5 | .FHJOB | All processes in the job |

Use of the superior process argument (.FHSUP) is legal only if the process has the superior process access capability (SC%SUP) enabled in its capability word. Meaningful operations may usually be performed with the top level process argument (.FHTOP) only if the process has WHEEL or OPERATOR capability enabled (SC%WHL or SC%OPR) in its capability word. Refer to Section 2.7.1 for information on the capability word.

Process handles in the range 400001 to 400777 are called relative process handles, and are generated by the monitor to refer to specific processes. (See the CFORK monitor call description.) These handles are valid only within the context of the process to which they are given. Thus, they may not be passed between processes. GFRKH may be used to convert process handles for use by another process.

1.2.8.1 Process/File Handle - Some monitor calls accept an 18-bit argument called a process/file handle. This handle is either a process handle (as defined in Section 1.2.8), or a JFN.

Note that string pointers and terminal identifiers cannot be used in this context. This is not a limitation, however, because the operations that use the process/file handle are used for changing page maps. Such operations are not meaningful for string pointers or terminals.

1.3 SYSTEM DATE AND TIME

The internal system date and time is a 36-bit quantity. It can be passed to a monitor call as an argument, or returned as a value. The internal date-and-time word has the following format:

day,,n

INTRODUCTION

where day is the number of days since November 17, 1858, and *n is the fractional part of the day elapsed since midnight, Greenwich Mean Time. n is the numerator of a fraction that has a denominator of 2**18. Thus the fraction

$$*n/2^{18}$$

represents the portion of the day elapsed since midnight. This format conforms to the Smithsonian Astronomical Date Standard.

Because the time is stored as Greenwich Mean Time, the monitor adds the value of the TIMEZONE offset to the internal date and time to obtain your local time. The TIMEZONE offset is specified in <SYSTEM>CONFIG.CMD. (See the Installation Guide for more information on the TIMEZONE offset.)

Monitor calls convert local dates and times to internal dates and times, and internal dates and times to local dates and times. Refer to Section 2.9.2 for more information about date and time conversion.

1.4 PROCESSING ERRORS

TOPS-20 provides a consistent way to handle all JSYS errors. Upon a successful return of most monitor calls, the instruction following the call is executed. If an error occurs during the execution of the call, the monitor examines the instruction following the call. If the instruction is a JUMP instruction with the AC field specified as 12-17, the monitor transfers control to a user-specified address. If the instruction is not a JUMP instruction, the monitor generates an illegal instruction trap indicating an illegal instruction, which the user's program can process via the software interrupt system (refer to Chapter 4 of the Monitor Calls User's Guide). If the user's program is not prepared to process the instruction trap, the program execution halts, and a message is output stating the reason for failure.

To place a JUMP instruction in his program, the user can include a statement using one of six predefined symbols. These symbols are:

| | | |
|--------|---------|---------------------|
| ERJMPR | address | (= JUMP 12,address) |
| ERCALR | address | (= JUMP 13,address) |
| ERJMPS | address | (= JUMP 14,address) |
| ERCALS | address | (= JUMP 15,address) |
| ERJMP | address | (= JUMP 16,address) |
| ERCAL | address | (= JUMP 17,address) |

and cause the assembler to generate a JUMP instruction. The JUMP instruction is a non-operation instruction (that is, a no-op) as far as the hardware is concerned. However, the monitor executes the JUMP instruction by transferring control to the address specified, which is normally the beginning of an error processing routine written by the

INTRODUCTION

user. If the user includes the ERJMP symbol, control is transferred as though a JUMPA instruction had been executed, and control does not return to his program after the error routine is finished. If the user includes the ERCAL symbol, control is transferred as though a PUSHJ 17, address instruction had been executed. If the error routine executes a POPJ 17, instruction, control returns to the user's program at the location following the ERCAL.

If the user includes the ERJMPR symbol, the monitor behaves the same as it would if the ERJMP symbol had been included, except that the last error encountered by the process is stored in the user's AC1. (Refer to Appendix B for the list of error codes, mnemonics, and message strings.) The ERCALR symbol functions the same as ERCAL except the error code encountered is returned in the user's AC1. ERJMPS and ERCALS function similarly except the monitor suppresses the storing of the error code in the user's AC1. Instead, AC1 is preserved and contains either the original contents from when the monitor call was given, or a partially updated value prior to the error.

Prior to the implementation of the ERJMP/ERCAL facilities, certain monitor calls returned control to the user's program at various locations after the calling address. Approximately one third of the JSYSs return to the +1 address only on failure, and to the location immediately following that (the +2 address) on successful execution of the call. A few calls return +1, +2, or +3, dependent on varying conditions of success or failure (for examples, see ERSTR% or GACTF%); and some calls do not return at all (see HALTF% or WAIT%). Refer to Chapter 3 the possible returns for each monitor call.

When a failure occurs during the execution of a monitor call, the monitor stores an error code. The error code indicates the cause of the failure. This error code is usually stored in the right half of AC1, but can also be stored in the monitor's data base or a user's data block. In either case, you can obtain the message associated with the error by using the GETER% or ERSTR% call.

The ERJMP/ERCAL facilities can also be used following a machine instruction, and will trap for the following conditions:

- o Illegal instruction
- o Illegal memory read
- o Illegal memory write
- o Pushdown list overflow

The ERJMP/ERCAL facilities can be used after all monitor calls, regardless of whether the call has one or two returns. To handle errors consistently, users are encouraged to employ either the ERJMPR, ERCALR, ERJMPS, or ERCALS symbol with all calls. All of the six predefined jump symbols are no-ops, unless they immediately follow a

INTRODUCTION

monitor call or instruction that fails. Error codes can be obtained by the program and translated into their corresponding error mnemonic and message strings with the GETER% and ERSTR% monitor calls.

TOPS-20 provides convenient macros and subroutines for handling monitor call error routines. They can be found in the system file MACSYM.MAC. Two such macros are EJSERR and EJSHLT. EJSERR prints out an error message and returns control to the next instruction following the failing monitor call. EJSHLT prints out an error message and halts processing of the program.

CHAPTER 2

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.1 ACCOUNTING FUNCTIONS

The monitor calls in this group initiate and delete jobs from the system. They also change and read accounting information about these jobs.

The following monitor calls perform accounting functions. Calls marked with an asterisk ("*") require privileges for specific functions.

| | |
|--------|---|
| GACCT* | Reads a file's account |
| GACTF | Reads a file's account |
| LOGIN | Logs a job into the system |
| SACTF | Sets a file's account |
| USAGE | Writes entries into the system's accounting data file |
| VACCT | Validates an account |

2.2 REFERENCING FILES

All files in the system, including the system's file directory, are normally referenced with the calls in this group. Section 2.11 describes the privileged calls for referencing the disk directly, without using the TOPS-20 file system.

2.2.1 File Specifications

A file in TOPS-20 is identified by its node name, device name, directory name, filename, file type, and generation number. These five items uniquely identify any file on the system that is accessible to a user. The device name identifies the device on which the file is stored. The directory name identifies the directory containing the file. The filename, type, and generation number identify a particular file in the directory.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

A file can also have attributes associated with it to further specify information about the file. See the description of the long-form GTJFN JSYS for a list of the possible file attributes.

The general format of a file specification is:

```
node::dev:<directory>name.typ.gen;attribute-1;attribute-2...
```

Refer to the TOPS-20 User's Guide for the complete description of file specifications.

If a field of the file specification (or filespec) is omitted, it can be supplied by the program or from standard system values. (Refer to Section 2.2.3.)

Whenever an ESC is encountered in the file specification string, the system looks for a file whose specification matches the fields input thus far. A match is indicated if the input string either exactly matches an entry in the appropriate table, or is an initial substring of exactly one entry. In the latter case, the portion of the matching entry not appearing in the input string is output to a specified output file. The field terminator is output also.

Recognition is done on successive fields with the fields being defaulted if need be. If the file specification cannot be uniquely determined, the system recognizes as many entire fields as are unique, and outputs a bell to the terminal, signifying that more input is required from the user.

CTRL/F behaves like ESC except recognition stops after the current field. This allows the filename to be recognized, for example, but not the file type.

If recognition is not used, then each field must be included as indicated in the general format above. The input must exactly match some existing file specification unless the program specifies in the GTJFN call that new specifications are allowed (output files).

Without ESC or CTRL/F, no recognition is done. The system substitutes the default values supplied by your program for fields completely omitted from the file specification. The file specification is complete whenever all fields have been recognized or a terminator has been input. File specification terminators are described in the GTJFN call description.

The following editing characters are recognized during the input of file specifications:

- DELETE erases one character. If no more characters remain in the input, a bell is output.

- CTRL/W deletes back to the last punctuation character. If no more characters remain in the input, a bell is output.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

CTRL/U aborts the entire filename-gathering operation.

CTRL/R retypes the entire input as specified so far and awaits further input.

2.2.2 Logical Names

Logical names are user-specified default values for one or more fields in a file specification. Through the use of logical names, the user can override standard file specification fields built into TOPS-20 programs because logical name fields take precedence over default fields set by a program. However, the user can still specify any fields explicitly since a logical name defines values to be used only if none are given by the user. The user defines logical names with the DEFINE command or the CRLNM monitor call. Refer to the TOPS-20 User's Guide for the complete description of logical names.

2.2.3 File Handles

It is necessary to have file handles that can be contained in a few bits and do not require extensive lookup procedures for each reference. The file specification is the fundamental handle on a file, but this specification fits neither criterion above. Therefore in TOPS-20, files are referenced by handles called JFNs (Job File Numbers). The JFN is a small number and is valid within the context of the job (that is, within any process of the job to which it is assigned). However, the handle is not valid between jobs. That is, JFN 2 in job 11 will generally be a handle on a completely different file than JFN 2 in job 18.

A JFN is associated with a file with either the GTJFN or GNJFN monitor call. The GTJFN call accepts a file specification and returns a JFN for the indicated file. If a field of the specification is omitted, it may be supplied by the program defaults or from standard system values. If the file specification refers to a group of files (because of wildcard characters, see below), the GNJFN call can be used to associate the JFN to the next file in the group.

A logical name can apply to one or more fields of the file specification passed to the GTJFN call. The logical name must be the first identifier passed to GTJFN and must be terminated with a colon.

The GTJFN call uses a certain search order when obtaining a field in a file specification. This order is as follows:

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

1. Use the field explicitly typed by the user or the one specified in the primary input string.
2. Use the value for the field that is specified in the logical name specification.
3. Use the value for the field that is specified in the default block by the program. This is only for the long form of the GTJFN call.
4. Use the system default value if all of the above searches fail.

In the special case of a device field specification, where the device name has been obtained from either the program default or the system default, the device field is checked to see if it is actually a logical name. If it is, then the values specified in its definition become defaults for all fields, including the device field.

If the specific call to GTJFN permits, wildcard characters (either an asterisk or a percent sign) can appear in the device, directory, filename, type, or generation number fields. (The percent sign cannot appear in the generation number field.) An asterisk matches any occurrence of the field, including a null field. An asterisk as part of a field matches 0 or more characters anywhere in the field. A percent sign matches any single existing character in the field. Upon completion of the operation, the JFN returned references the first file found when scanning in the following order:

- In order by structure name
(PS: is first, arbitrary order for others)
- In alphabetic order by directory name
- In alphabetic order by filename
- In alphabetic order by file type
- In ascending numeric order by generation number

Note that for structures, only the construct DSK*:
can be used. This means all available structures on the system.

The GNJFN call can then be given to associate the JFN to the next file that matches the file specification.

The fullword JFN (flags,,JFN) is termed an "indexable file handle" because it accepts a generic file specification (one including wildcard characters) and can be successively associated (by GNJFN) with each file matching the specification. Thus the JFN is "indexed" through a range of files.

The number and type of files in the range are limited by the file specification, the privileges of the program, and the protection of individual files and directories within the file system. A program with WHEEL capabilities enabled can access any file in the TOPS-20 file system.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The maximum number of JFN's allowed depends upon the space reserved for JFN-related information in the Job Storage Block (JSB). Currently the maximum number of JFN's allowed is 140 (octal).

The JFN's 100 (.PRIIN) and 101 (.PRIOU) are reserved for the primary input and output designators, respectively, and are never returned by the GTJFN (or GNJFN) call. The JFN 377777 (.NULIO) is reserved for the null designator.

Ordinarily, the process of getting a file handle with GTJFN consists of the following:

1. The user specifies the file name string.
2. GTJFN checks the file name string for grammatical correctness.
3. GTJFN checks the file for validity (For example, does the file actually exist?)
4. If the file name passes these two checks, GTJFN returns a JFN or handle for the file.

Thus a JFN is associated with an actual file in the TOPS-20 file system.

It is sometimes desirable to skip the step of checking a JFN for validity. This is necessary any time that the association between the JFN and the physical file cannot be made, as happens when a JFN is requested for a file on magnetic tape. Also, it may be that the user himself wishes to prevent the JFN/file association from being made in order to check the file specification for grammatical correctness and then manipulate the file specification by adding or removing selected fields, or comparing it against another file specification. This type of JFN is termed a "parse-only" JFN. As it is not associated with any file, no file operations may be performed on it.

Only the following JSYSs accept a parse-only JFN:

1. JFNS - converts a JFN to its file specification (in characters)
2. WILD% - compares character strings and file specifications

2.2.4 File References

All file operations are initiated by acquiring a JFN on a file using the GTJFN (or GNJFN) call. Some file operations, such as deleting,

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

renaming, and status queries about the file, may be performed immediately after the JFN is acquired. Certain operations, particularly data transfers, require that the file be opened with an OPENF call on the JFN.

When the user opens a file, he specifies the byte size to be used for byte I/O operations and the access requested to the file. Several implicit initialization operations, which affect subsequent references to the file, are also invoked when a file is opened. For example, a file's position pointer is normally reset to the beginning of the file such that the first sequential input operation reads the beginning data of the file.

Access to files on regulated structures (those being tracked by the accounting system) cannot be given until the mount count for that structure is incremented with the .MSIMC function of the MSTR JSYS (or with the TOPS-20 MOUNT STRUCTURE command). All JFN's must be released before the mount count can be decremented with the .MSDMC function of the MSTR JSYS (or the TOPS-20 DISMOUNT STRUCTURE command).

All structures are regulated by default except the primary structure (PS:).

2.2.4.1 Files and Devices - Under TOPS-20, most devices may be treated as if they were files. For example, a GTJFN, OPENF, CLOSF, etc. may be performed directly on magnetic tape device MT1: without specifying a file name. This is because the device name itself is the file name. Disk devices, however, have multiple directories and multiple files, and the device name itself is not sufficient to uniquely identify a file. The general rule is that, for a complete TOPS-20 file specification, only those fields necessary to make the file unique for that device are required to get a JFN for the file. Thus, for most devices, the device name itself is sufficiently unique to get a JFN for the file. In this manual, when the phrase "opening a device" is used, it is in reference to the feature described above.

For TOPS-20, disk devices are the only major exception to the rule that devices can be treated as files. Labeled tapes on MT: devices may be referenced either by device name alone (which gives access to all files on the tape) or by device name and file name (which gives access only to the specified file).

2.2.5 Sample Program

The following sample program acquires JFN's, opens both an input and an output file, and then copies data from the input file to the output file in 7-bit bytes until the end of the input file is encountered.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

```

;*** PROGRAM TO COPY INPUT FILE TO OUTPUT FILE. ***
;   (USING BIN/BOU AND IGNORING NULL'S)

        TITLE FILEIO           ;TITLE OF PROGRAM
        SEARCH MONSYM,MACSYM    ;SEARCH SYSTEM JSYS-SYMBOL
                                   ;LIBRARIES

;*** IMPURE DATA STORAGE AND DEFINITIONS ***

INJFN:  BLOCK 1                ;STORAGE FOR INPUT JFN
OUTJFN: BLOCK 1                ;STORAGE FOR OUTPUT JFN

        PDLEN=3                ;STACK HAS LENGTH 3
PDLST:  BLOCK PDLEN            ;SET ASIDE STORAGE FOR STACK
STDAC.  ;DEFINE STANDARD JSYS ACs

;*** PROGRAM INITIALIZATION ***

START:  RESET%                 ;CLOSE FILES AND INITIALIZE PROCESS
        MOVE P,[IOWD PDLEN,PDLST] ;ESTABLISH STACK

;*** GET INPUT-FILE ***

INFIL:  HRROI T1,[ASCIZ /
INPUT FILE: /]                 ;PROMPT FOR INPUT FILE
        PSOUT%                 ;ON CONTROLLING TERMINAL
        MOVX T1,GJ%OLD+GJ%FNS+GJ%SHT;SEARCH MODES FOR GTJFN
                                   ;[EXISTING FILE ONLY , FILE-NR'S IN B
                                   ; SHORT CALL ]

        MOVE T2,[.PRIIN,,.PRIOU] ;GTJFN'S I/O WITH
                                   ; CONTROLLING TERMINAL
        GTJFN%                 ;GET JOB FILE NUMBER (JFN)
        ERJMP [ PUSHJ P,WARN    ;IF ERROR, GIVE WARNING
                JRST INFIL]    ;AND LET HIM TRY AGAIN
        MOVEM T1,INJFN         ;SUCCESS, SAVE THE JFN

;*** GET OUTPUT-FILE ***

OUTFIL: HRROI T1,[ASCIZ /
OUTPUT FILE: /]                 ;PROMPT FOR OUTPUT FILE
        PSOUT%                 ;PRINT IT
        MOVX T1,GJ%FOU+GJ%MSG+GJ%CFM+GJ%FNS+GJ%SHT ;GTJFN
                                   ; SEARCH MODES [DEFAULT TO NEW
                                   ; GENERATION , PRINT MESSAGE ,
                                   ; REQUIRE CONFIRMATION
                                   ; FILE-NR'S IN B , SHORT CALL ]

        MOVE T2,[.PRIIN,,.PRIOU] ;I/O WITH CONTROLLING TERMINAL
        GTJFN%                 ;GET JOB-FILE NUMBER
        ERJMP [ PUSHJ P,WARN    ;IF ERROR, GIVE WARNING
                JRST OUTFIL]   ;AND LET HIM TRY AGAIN

```

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

```

MOVEM T1,OUTJFN          ;SAVE THE JFN

;NOW, OPEN THE FILES WE JUST GOT

; INPUT

MOVE T1,INJFN           ;RETRIEVE THE INPUT JFN
MOVX T2,FLD(7,OF%BSZ)+OF%RD      ;DECLARE MODES FOR OPENF
                                   ;[7-BIT BYTES + INPUT]
OPENF%                   ;OPEN THE FILE
ERJMP FATAL              ;IF ERROR, GIVE MESSAGE AND STOP

; OUTPUT

MOVE T1,OUTJFN          ;GET THE OUTPUT JFN
MOVX T2,FLD(7+OF%BSZ)+OF%WR      ;DECLARE MODES FOR OPENF
                                   ;[7-BIT BYTES + OUTPUT]
OPENF%                   ;OPEN THE FILE
ERJMP FATAL              ;IF ERROR, GIVE MESSAGE AND STOP

;*** MAIN LOOP :COPY BYTES FROM INPUT TO OUTPUT ***

LOOP: MOVE T1,INJFN      ;GET THE INPUT JFN
      BIN%               ;TAKE A BYTE FROM THE SOURCE
      ERJMP DONE        ;IF ERROR, CHECK FOR END OF FILE.
      JUMPE T2,LOOP     ;SUPPRESS NULLS
      MOVE T1,OUTJFN    ;GET THE OUTPUT JFN
      BOUT%             ;OUTPUT THE BYTE TO DESTINATION
      ERJMP FATAL       ;IF ERROR, GIVE MESSAGE AND STOP
      JRST LOOP        ;LOOP, STOP ONLY ON A 0 BYTE
                        ;(FOUND AT LOOP+2)

;*** TEST FOR END OF FILE, ON SUCCESS FINISH UP ***

DONE:  GTSTS%            ;GET THE STATUS OF INPUT FILE.
      TXNN T2,GS%EOF    ;AT END OF FILE?
      PUSHJ P,FATAL    ;NO, I/O ERROR

CLOSIF: MOVE T1,INJFN   ;YES, RETRIEVE INPUT JFN
      CLOSF%           ;CLOSE INPUT FILE
      ERJMP FATAL      ;IF ERROR, GIVE MESSAGE AND STOP

CLOSOFF: MOVE T1,OUTJFN ;RETRIEVE OUTPUT JFN
      CLOSF%          ;CLOSE OUTPUT FILE
      ERJMP FATAL     ;IF ERROR, GIVE MESSAGE AND STOP
      HRROI T1,[ASCIZ/
[DONE]/]                ;SUCCESSFULLY DONE
      PSOUT%          ;PRINT IT
      JRST ZAP        ;STOP

;*** ERROR HANDLING ***

```

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

```
FATAL:  HRROI T1,[ASCIZ/  
?/]                                ;FATAL ERRORS PRINT ? FIRST  
      PUSHJ P,ERROR                  ;THEN PRINT ERROR MESSAGE,  
      JRST ZAP                       ;AND STOP  
  
WARN:   HRROI T1,[ASCIZ/  
%/]                                ;WARNINGS PRINT % FIRST AND FALL  
                                           ; THRU 'ERROR' BACK TO CALLER  
  
ERROR:  PSOUT%                      ;PRINT THE ? OR %  
      MOVE T1,[.PRIOU]              ;DECLARE PRINCIPAL OUTPUT DEVICE  
                                           ;FOR ERROR MESSAGE  
  
      MOVE T2,[.FHSLF,, -1]         ;CURRENT FORK,, LAST ERROR  
      SETZB T3,T4                   ;NO LIMIT,, FULL MESSAGE  
      ERSTR%                         ;PRINT THE MESSAGE  
      JFCL                           ;IGNORE UNDEFINED ERROR NUMBER  
      JFCL                           ;IGNORE ERROR DURING EXECUTION  
                                           ;OF ERSTR  
      POPJ P,                       ;RETURN TO CALLER  
  
ZAP:    HALTF%                      ;STOP  
      JRST START                    ;WE ARE RESTARTABLE  
      END START                     ;TELL LINKING LOADER  
                                           ;START ADDRESS
```

2.2.6 File Access

TOPS-20 provides a general mechanism for protecting files against unauthorized access. This mechanism includes the ability to protect access to files on a directory-wide basis as well as on an individual-file basis.

Generally, access to a file depends on the kind of access desired and the relationship of the user making the access to the directory containing the file. The possible relationships a user may have to the file's directory are:

1. The directory containing the file is the user's connected or one of the user's accessed directories. Users satisfying this relationship have owner access to the files in the directory.
2. The directory containing the file is in the same group as the user. Users satisfying this relationship have group member access to the files in the directory.
3. The directory containing the file is outside the group membership. Users satisfying this relationship have world access to the files in the directory.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Both users and directories may belong to groups. The group-member relationship is satisfied if both the directory and the user belong to one or more of the same groups. Groups are assigned by the system manager or operator. (Refer to the TOPS-20 System Manager's Guide.)

The type of access permitted to a file for each relationship is represented by the value of a 6-bit field. The possible values are:

| Value | Symbol | Meaning |
|-------|--------|--|
| 40 | FP%RD | Read access |
| 20 | FP%WR | Write access |
| 10 | FP%EX | Execute access |
| 4 | FP%APP | Append access |
| 2 | FP%DIR | Directory listing access. If a user does not have at least this type of access, a GTJFN will find the file only if wildcards are not used. A GNJFN will not find the file. |

The following table illustrates some useful combinations of the values shown above:

| Value | Symbol | Meaning |
|-------|--------------|--|
| 12 | FP%EX+FP%DIR | Execute-only access |
| 42 | FP%RD+FP%DIR | Usual protection allowing users to access a file without being able to modify it. |
| 60 | FP%RD+FP%WR | Good for hiding files that specific programs can write to. Programs should be execute-only and the program should set the "restricted" access bit in the GTJFN so as not to reveal the filename. |

The 6-bit field and the three relationships (owner, group, remaining users) are represented by an 18-bit code, with bits 0-5 being the owner, bits 6-11 being the group, and bits 12-17 being the remaining users. When a particular bit is on, the corresponding access is permitted for the particular relationship.

The access given to a group member includes the access given to all members outside the group. Also, the access given to the owner includes the access given to group members. Thus, the owner of a file or a user in the owner's group cannot have less access than users outside the group.

2.2.7 Directory Access

Access to a directory is protected in a manner similar to, but distinct from, that of a file. An 18-bit code, containing three 6-bit fields, is associated with each directory. Each of the three fields

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

controls access by users in the same way that access to files is controlled. For directories, however, each 6-bit field can have one of the following values.

| Value | Symbol | Meaning |
|-------|--------|---|
| 40 | DP%RD | Accessing files in the directory according to the access code on the individual files is allowed. A GTJFN call for a file in the directory will fail if the user does not have this access. |
| 10 | DP%CN | Connecting to the directory without giving a password is allowed. With this access, a group member can change the FDB (as the owner) as well as times, dates, and accounting information for files in the directory. Other operations on the files are subject to the access codes of the files. If the user is connected to the directory, he has ownership access to the files; if he is not connected, he has group membership access. |
| 4 | DP%CF | Creating files in the directory is allowed. |

When a user requests access to a file, the monitor checks the directory access code first. If the directory code allows the desired access, the monitor then checks the access code of the individual file.

The access actually granted to a file is specified when the user opens the file with the OPENF call. If the access specified in the OPENF call is the same as or less than the access permitted by the 18-bit access code, the user is granted access to the file. Thus, for a user to be granted access to a specific file, two conditions must be met:

1. The access code (both directory and file) must permit the user to access the file in the desired manner (for example, read, write).
2. The file must not be open for a conflicting type of access.

2.2.8 File Descriptor Block

Each file has an associated File Descriptor Block (FDB) that contains various information about the file. The format of the FDB is shown in Table 2-1.

The description of each word or bit in the FDB indicates whether the user can change it, and if so, what types of access are required. The types of access are:

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

1. WRITE - write access
2. OWNER - owner access
3. W/OPR - WHEEL or OPERATOR capabilities enabled

In some cases, separate JSYSs are required to read, set, and/or clear various words or bits. These functions are indicated by:

1. (R) - read
2. (S) - set
3. (C) - clear
4. (SC) - set/clear

Table 2-1: File Descriptor Block (FDB)

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .FBHDR | FDB header word. Individual fields are as follows: B0-B28 Reserved for DIGITAL. UNCHANGABLE B29-35(FB%LEN) Length of this file's FDB UNCHANGABLE |
| 1 | .FBCTL | B0(FB%TMP) File is temporary. JSYS WRITE OWNER W/OPR CHFDB N Y Y B1(FB%PRM) File is permanent. The contents of the file may be deleted, but the FDB may not. JSYS WRITE OWNER W/OPR CHFDB N Y Y |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

B2(FB%NEX) File does not yet have a file type;
file does not really exist.

UNCHANGEABLE

B3(FB%DEL) File is deleted.

JSYS WRITE OWNER W/OPR

CHFDB N Y* Y

*This bit may be changed by the
owner providing that bit FB%ARC (in
.FBCTL) is not set.

B4(FB%NXF) File does not exist because it has
not yet been closed.

UNCHANGEABLE

B5(FB%LNG) File is longer than 512 pages.

UNCHANGEABLE

B6(FB%SHT) Reserved for DIGITAL.

UNCHANGEABLE

B7(FB%DIR) File is a directory.

UNCHANGEABLE

B8(FB%NOD) File is not to be saved by the
backup system.

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

B9(FB%BAT) File may have one or more bad pages.
This bit indicates that I/O errors
have occurred for a page (or pages)
of a file and the contents of these
pages are suspect.

This bit is set whenever the system
has a disk I/O error on a page of an
open file. The faulty disk address
is also added to the list in the
system's BAT blocks for that disk
structure.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

If an EXPUNGE is performed for a file for which bit FB%BAT is set, the system performs an additional function as it releases the pages of the file back to the available resource pool: it checks each disk address in the file against the list of bad regions in the structure's BAT blocks and if it finds a match, it leaves that page marked as "in use" in the bit map of available disk pages, so that the faulty page is not reused.

UNCHANGEABLE

B10(FB%SDR) Directory has subdirectories.

UNCHANGEABLE

B11(FB%ARC) File has archive status. Appropriate words in the FDB (below) specify where the file is archived.

| JSYS | WRITE | OWNER | W/OPR |
|------|-------|-------|-------|
|------|-------|-------|-------|

| | | | |
|------|---|---|---|
| ARCF | N | N | Y |
|------|---|---|---|

B12(FB%INV) File is invisible. Invisible files can be seen only by using the G1%IIN option to GTJFN.

| JSYS | WRITE | OWNER | W/OPR |
|------|-------|-------|-------|
|------|-------|-------|-------|

| | | | |
|-------|---|---|---|
| CHFDB | N | Y | Y |
|-------|---|---|---|

B13(FB%OFF) File is offline. This is set by DELF when it removes the contents from disk and cleared when ARCF restores the contents to disk.

| JSYS | WRITE | OWNER | W/OPR |
|------|-------|-------|-------|
|------|-------|-------|-------|

| | | | |
|---------|---|---|---|
| DELF(S) | N | N | Y |
|---------|---|---|---|

| | | | |
|---------|---|---|---|
| ARCF(C) | N | N | Y |
|---------|---|---|---|

B14-B17(FB%FCF)

File class field. If value of field is 0(.FBNRM), file is not an RMS file. If value of field is 1(.FBRMS), file is an RMS file.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | | | |
|---|-------------|--|-------|-------|-------|
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | Y | Y | Y |
| | B18(FB%NDL) | Do not delete this file. Do not delete even if overwritten by a write or a rename. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | N | Y |
| | B19(FB%WNC) | Last write not closed. File has not been closed by all writers. Page count may be incorrect. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | N | Y |
| | B20(FB%FOR) | File has FORTRAN-style line printer carriage control characters. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | Y | Y | Y |
| | B21(FB%SEC) | File is secure. | | | |
| 2 | .FBEXL | Link to FDB of next file with the same name but different file type. | | | |
| | | UNCHANGEABLE | | | |
| 3 | .FBADR | Disk address of file index block. | | | |
| | | UNCHANGEABLE | | | |
| 4 | .FBPRT | File access code. LH: 500000 | | | |
| | | UNCHANGEABLE | | | |
| | | RH: file access bits. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | Y | N |
| 5 | .FBCRE | Date and time that the file was closed after the last write to the file. Modified when any program writes to the file. | | | |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | | | |
|----|--------|--|-------|-------|-------|
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | N | Y |
| 6 | .FBAUT | Pointer to string containing the name of the author. This word is not under direct user control. It is only changed indirectly, when the file author string is changed. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | GFUST(R) | Y | Y | Y |
| | | SFUST(SC) | N | Y | N |
| 7 | .FBGEN | Generation and directory numbers of file. | | | |
| | | LH(FB%GEN): generation number of the file. | | | |
| | | UNCHANGABLE | | | |
| | | RH(FB%DRN): monitor internal directory number of the file (only if B7 of .FBCTL is on). | | | |
| | | UNCHANGABLE | | | |
| 10 | .FBACT | Account information. This word contains a byte pointer to an alphanumeric account designator; it can be changed with the SACTF monitor call. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | SACTF | Y | Y | Y |
| 11 | .FBBYV | File I/O information. | | | |
| | | B0-B5(FB%RET) | | | |
| | | Number of generations to retain (retention count). If two generations of the same file have different retention counts, the count is taken from the generation currently being used. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | Y | Y | Y |
| | | B6-B11(FB%BSZ) | | | |
| | | File byte size. This field can be changed by user with write access. | | | |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

B14-B17(FB%MOD)

Data mode of last open of file.
This field can be changed by user
with write access.

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

B18-B35(FB%PGC)

Page count of file. Note that the
monitor keeps the page count
updated, so under normal
circumstances a user need not and
should not alter this count.

JSYS WRITE OWNER W/OPR

CHFDB N N Y

12 .FBSIZ Number of bytes in the file. (Refer to Section 2.2.11.)

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

13 .FBCRV Date and time of creation of file.

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

14 .FBWRT Date and time that the file was opened when the last write to the file was made.

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

15 .FBREF Date and time of last nonwrite access to file.

JSYS WRITE OWNER W/OPR

CHFDB Y Y Y

16 .FBCNT Count word.
LH: number of writes to file.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | | | |
|----|--------|---|-------|-------|-------|
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | N | Y |
| | | RH: number of references to file. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | N | Y |
| 17 | .FBBK0 | Used by DUMPER for backup purposes. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | CHFDB | N | N | Y |
| 20 | .FBBK1 | Reserved for DEC. | | | |
| | | UNCHANGABLE | | | |
| 21 | .FBBK2 | Reserved for DEC | | | |
| | | UNCHANGABLE | | | |
| 22 | .FBBBT | The right half contains the number of pages in the file when the contents were deleted from disk. | | | |
| | | UNCHANGABLE | | | |
| | | The left half is used for the following flags: | | | |
| | | B1(AR%RAR) User request for a file to be archived. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | Y | Y | Y |
| | | B2(AR%RIV) System request for an involuntary migration of a file. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | N | Y |
| | | B3(AR%NDL) Do not delete the contents of the file from disk when the archival is complete. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | Y | Y |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

B4(AR%NAR) Resist involuntary migration. This bit is a note from the user to the system access control program asking that the file not be moved offline if possible.

| | | | |
|------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| ARCF | N | Y | Y |

B5(AR%EXM) File is exempt from involuntary migration.

| | | | |
|------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| ARCF | N | N | Y |

B6(AR%1ST) First pass of an archival-collection run is in progress.

| | | | |
|-------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| CHFDB | N | N | Y |

B7(AR%RFL) Restore failed. Set by ARCF to indicate that the restore it is waiting for has failed.

| | | | |
|------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| ARCF | N | N | Y |

B10(AR%WRN) Generate a message warning that the file's off-line expiration date is approaching.

7B17(AR%RSN) Reason file was moved offline:

- .AREXP(1) file expired
- .ARRAR(2) archiving was requested
- .ARRIR(3) migration was requested

| | | | |
|----------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| ARCF(W) | N | N | Y |
| GTFDB(R) | Y | Y | Y |

B18-B35(AR%PSZ) The right half of .FBBBT is used to

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

store the number of pages in a file when the contents were removed from disk.

| | | | |
|----------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| ARCF(W) | N | N | Y |
| GTFDB(R) | Y | Y | Y |

23 .FBNET On-line expiration date and time. Specifies the date and time at which a file is considered expired, or specifies an interval (in days) after which the file is considered expired.

| | | | |
|-------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| SFTAD | N | Y | Y |

24 .FBUSW User-settable word.

| | | | |
|-------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| CHFDB | N | Y | Y |

25 .FBGNL Address of FDB for next generation of file.

UNCHANGABLE

26 .FBNAM Pointer to filename block.

UNCHANGABLE

27 .FBEXT Pointer to file type block.

UNCHANGABLE

30 .FBLWR Pointer to string containing the name of the user who last wrote to the file. This name is read with the GFUST monitor call and can be changed with the SFUST monitor call.

Note that word .FBLWR may only be changed indirectly (by specifying a new name string). This word cannot be changed directly.

| | | | |
|-----------|-------|-------|-------|
| JSYS | WRITE | OWNER | W/OPR |
| GFUST(R) | Y | Y | Y |
| SFUST(CS) | N | N | Y |

31 .FBTDT Archive or collection tape-write date and time. This is the date and time (in internal format)

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

that file was last written to tape (for either archiving or migration).

| | | | | | |
|----|--------|--|-------|-------|-------|
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | N | Y |
| 32 | .FBFET | Offline expiration date and time. Specifies the date and time (or interval) after which a file in the archives or on virtual disk is considered expired. Used for tape recycling. Modified by SFTAD. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | SFTAD | Y | Y | Y |
| 33 | .FBTP1 | Contains the tape ID for the first archive or collection run. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | N | Y |
| 34 | .FBSS1 | Contains the saveset and tape file numbers for the first tape. The left half is the number of the saveset in which the file is recorded, and the right half is the tape file number within that saveset. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | N | Y |
| 35 | .FBTP2 | Tape ID for second archive or collection run. Otherwise similar to .FBTP1. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | N | Y |
| 36 | .FBSS2 | Saveset and tape file numbers for the second archive or collection run. Otherwise similar to .FBSS1. | | | |
| | | JSYS | WRITE | OWNER | W/OPR |
| | | ARCF | N | N | Y |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The maximum length FDB block that TOPS-20 will create (37 octal) may be specified with the symbol .FBLLEN.

2.2.9 Primary Input and Output Files

Each process in a job has a primary input file and a primary output file. Both files are normally the controlling terminal, but can be changed to other files (with the SPJFN call).

The primary input and output files are referenced with designators .PRIIN (JFN 100) and .PRIOU (JFN 101), respectively. Programs should be coded to do their "terminal" I/O to these designators, so that they can be used with command files without modification. Only in extreme cases should a program reference its controlling terminal (.CTTRM) directly.

2.2.10 Methods of Data Transfer

The most simple form of I/O is sequential byte I/O, as shown in the sample program. (Refer to Section 2.2.5.) This form of data transfer may be used with any file. A pointer maintained in the monitor is implicitly initialized when a file is opened and advanced as data is transferred. For files on disk, there are two other methods of data transfers. First, random access byte I/O is possible by using the SFPTR call or the RIN/ROUT calls. Second, entire pages of data may be mapped with the PMAP call.

2.2.11 File Byte Count

For disk files, TOPS-20 maintains a file byte count (.FBSIZ) in the FDB. This count is set by the monitor when sequential output (for example, BOUT, SOUT) occurs to the file and thus, on sequential output, reflects the number of bytes written in the file.

When output occurs to the file using the PMAP call, the monitor does not set the file byte count. In this case, the number of bytes in the file may be different from the file byte count stored in the FDB. To allow sequential I/O to occur later to the file, the program should update the file byte count (.FBSIZ) and the file byte size (FB%BSZ) in the FDB before closing the file. This is done with the CHFDB monitor call.

When output occurs to the file using random output calls (ROUT, for example), the file byte count is a number one greater than the highest byte number in the file. The file byte count is interpreted according to the byte size stored in the FDB, not the byte size specified when the file is opened. When a new file is opened, the byte size stored

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

in the FDB is 36 bits, regardless of the byte size specified in the OPENF call. If the program executes a CHFDB call to change the file byte count, it must usually change the byte size (FB%BSZ) so that both values reflect the same size bytes.

2.2.12 EOF Limit

There is an EOF limit associated with every opening of a file. This limit is the number of bytes that can be read with a sequential input call (for example, BIN, SIN). When the program attempts to read beyond this limit using sequential input, the call returns a 0 byte and an end-of-file condition. This condition may generate a software interrupt (refer to Section 2.6) if the user has not included an ERJMP or ERCAL as the next instruction following the call. (Refer to Chapter 1.)

The EOF limit is computed when the file is opened with the OPENF call. The monitor computes this limit by determining the total number of words in the file and dividing this number by the byte size given in the OPENF call. The total number of words in the file is determined from the file byte count (.FBSIZ) and the file byte size (FB%BSZ) stored in the FDB.

Note that page-mode I/O JSYSSs, such as PMAP, ignore the EOF limit and can read any existing page of the file. However, page-mode JSYSSs can only read pages within an existing file section (the address space of a file delimited by 1 index block - 512 pages).

2.2.13 Input/Output Errors

While performing I/O or I/O-related operations, it is possible to encounter one or more error conditions. Some of these are user-caused errors (for example, illegal access attempts), and others are I/O device or medium errors. TOPS-20 indicates such error conditions by setting error bits in the JFN status word (refer to the GTSTS call) and by initiating a software interrupt request (refer to Section 2.6) if the user has not included an ERJMP or ERCAL after the call. If the process in which an I/O error occurs is not prepared to process the interrupt, the interrupt is changed into a process terminating condition with the expectation that the process' immediate superior will handle the error condition. The TOPS-20 Command Language is prepared to detect and diagnose I/O errors; thus, a process running directly beneath the process containing the Command Language need not do its own I/O error handling unless it chooses to do something special.

I/O errors can occur while a process is executing ordinary machine instructions as well as JSYSSs. For example, if a PMAP operation is

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

performed that maps a page of a file into a page of a process, the file I/O transfer does not usually occur until a reference is made by the process to that particular page of the file. If there is an I/O error in the transfer, it is detected at the time of this reference.

An attempt to do I/O to a terminal that is assigned to another job (as a controlling terminal or with the ASND call) normally results in an error, but is legal if the process has the WHEEL capability enabled.

2.2.13.1 Testing for End-of-File - The GTSTS JSYS, used in conjunction with ERCAL (or ERJMP), is used to test for end-of-file. The following code fragment illustrates this:

```
        MOVE      T1,INJFN      ;Get input JFN
        BIN%      ;Read a byte
        ERCAL EOFSTST
        .
        .                      ;Process byte
        .
EOFSTST: MOVE      T1,INJFN      ;Get input JFN
        GTSTS%    ;Get status of that JFN
        TXNN      T2,GS%EOF      ;Did end of file occur?
        PUSHJ     P,FATAL        ; No, I/O error occurred
        MOVE      T1,INJFN      ; Yes, close file
        CLOSF%
        ERCAL     FATAL          ;If can't close, issue message
        POPJ      P,            ;OK to return

FATAL:  .                      ;Here to issue error messages
        .                      ; on fatal file errors
        .
        HALTF%    ;Halt on fatal error
```

In the example above, the ERCAL after the BIN is executed only if a file error condition arises. The code that is entered as a result of the ERCAL can then do a GTSTS for the appropriate file and test for end-of-file.

An alternate method to test for end-of-file is to use the GETER JSYS and determine if the last error for the process is IOX4 (end of file reached).

The following monitor calls used in referencing files (including I/O functions). Calls marked with an asterisk ("*") require privileges for specific functions.

| | |
|--------|---------------------------------|
| ACCES* | Specifies access to a directory |
| BIN | Reads the next byte |
| BKJFN | Backspaces file's pointer |
| BOU | Writes the next byte |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | |
|--------|---|
| CHFDB* | Changes a File Descriptor Block |
| CHKAC | Checks access to a file |
| CLOSF | Closes a file |
| CLZFF | Closes a process's files |
| CRDIR* | Creates or modifies a directory |
| CRLNM* | Creates a logical name |
| DELDF* | Expunges deleted files |
| DELF* | Deletes a file |
| DELNF | Retains specified number of generations of file |
| DIRST | Translates directory or user number to a string |
| DUMPI | Reads data in unbuffered data mode |
| DUMPO | Writes data in unbuffered data mode |
| FFFFP | Finds first free file page |
| FFUFD | Finds first used file page |
| FLIN | Reads a floating-point number |
| FLOUT | Writes a floating-point number |
| GACTF | Reads a file's account |
| GFUST | Reads the author or last writer name string |
| GNJFN | Assigns a JFN to the next file |
| GPJFN | Returns primary JFN's |
| GTfDB | Reads a File Descriptor Block |
| GTJFN | Assigns a JFN to a file |
| GTSTS | Reads file's status |
| INLNM | Writes logical names |
| JFNS | Translates a JFN to a string |
| LNMST | Translates logical name to string |
| MRECV* | Retrieves IPCF message |
| MSEND* | Sends IPCF message |
| MSTR* | Performs structure-related functions |
| MUTIL* | Performs IPCF functions |
| NIN | Reads a number |
| NOU | Writes a number |
| OPENF | Opens a file |
| PBIN | Reads byte from primary input designator |
| PBOU | Output byte to primary output designator |
| PMAP | Maps pages |
| PSOU | Writes string to primary output designator |
| QUEUE% | Communicates with spooling system and operator |
| RCDIR | Translates directory name to number |
| RCUSR | Translates user name to number |
| RCVIN% | Receives an Internet message |
| RDTTY | Reads data from primary input designator |
| RFBSZ | Reads file's byte size |
| RFPTR | Reads file's pointer |
| RFTAD | Reads file's time and dates |
| RIN | Reads a byte nonsequentially |
| RLJFN | Releases a JFN |
| RNAMF | Renames a file |
| ROU | Writes a byte nonsequentially |
| RSCAN | Reads and outputs rescan buffer |
| SACTF | Sets a file's account |
| SFBSZ | Sets file's byte size |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | |
|--------|--|
| SFPTR | Sets file's pointer |
| SFTAD* | Sets file's time and dates |
| SFUST* | Changes the author or last writer name string |
| SIN | Reads a string |
| SINR | Reads a record |
| SIZEF | Obtains file's length |
| SMAP% | Maps sections |
| SNDIN% | Sends an Internet message |
| SPJFN | Sets primary JFN's |
| SOUT | Writes a string |
| SOUTR | Writes a record |
| STI* | Simulates terminal input |
| STSTS | Sets file's status |
| SWJFN | Transposes two JFN's |
| TEXTI | Reads data from terminal or file |
| TTMSG* | Sends message to terminal(s) |
| UFPGS | Updates file's pages |
| WILD% | Compares a wild file specification against a non-wild file specification. Also compares strings. |

2.3 OBTAINING INFORMATION

The monitor calls in this group are used to obtain information from the system, such as the time of day, resources used by the current job, error conditions, and the contents of system tables.

Several of these calls return time values (intervals and accumulated times, for example). Unless otherwise specified, these values are integer numbers in units of milliseconds.

2.3.1 Error Mnemonics and Message Strings

Each failure for a JSYS is associated with an error number identifying the particular failure. These error numbers are indicated in the manual by mnemonics (DEVX1, for example), and are listed with the appropriate calls.

Some calls return the error number in the right half of an accumulator, usually in AC1; however, all calls leave the number in the Process Storage Block for the process in which the error occurred. Thus, a process can obtain the number for the last error that occurred (by means of the GETER call).

In addition to the mnemonic of six characters or less, each error number has a text message associated with it that describes the error in more detail. The ERSTR call can be used to return the message string associated with any given error number. This call should be used for handling error returns.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Refer to Chapter 3 and Appendix B for the listing of the error numbers, mnemonics, and messages.

2.3.2 System Tables

The contents of several system tables are available to programs for such purposes as generating status reports and collecting system performance statistics. Each table is identified by a fixed name of up to six characters, and consists of a variable number of entries. The -1 entry in each table is the negative of the number of data entries in the table; the data entries are identified by an index that increments from 0.

Two calls exist for accessing tables. The first, SYSGT, accepts a table name and returns the table length, its first data entry, and a number identifying the table. The second, GETAB, accepts the table number returned by SYSGT, or obtained from the MONSYM file, and returns additional entries from the table.

The system tables are as follows. Numeric table indexes are given in octal. Parallel tables, those for which a given index produces related information, are indicated by "(Pn)" where n is a unique number for that set of parallel tables.

Table 2-2: System Tables

| Name | Index | Contents |
|--------|-------|--|
| APRID | | Processor serial number |
| ACTJOB | | Range of active jobs on the system from lowest job in use to highest job in use (not including Job 0). |
| BLDTD | | Date and time system was generated |
| CSTAT | | CI statistics table |
| | 0 | CI packets sent |
| | 1 | CI packets received |
| | 2 | SCA overhead messages sent |
| | 3 | SCA overhead messages received |
| | 4 | MSCP driver messages sent |
| | 5 | MSCP driver messages received |
| | 6 | MSCP server messages sent |
| | 7 | MSCP server messages received |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|--------|--------|---|
| | 10 | CFS messages sent |
| | 11 | CFS messages received |
| | 12 | SCS% messages sent |
| | 13 | SCS% messages received |
| | 14 | CI command queue 0 |
| | 15 | CI command queue 1 |
| | 16 | CI command queue 2 |
| | 17 | CI command queue 3 |
| | 20 | IP datagrams sent |
| | 21 | IP datagrams received |
| | 22 | DECnet datagrams sent |
| | 23 | DECnet datagrams received |
| | 24 | SCS% datagrams sent |
| | 25 | SCS% datagrams received |
| | 26 | MSCP driver datagrams received |
| | 27 | HSCP error-log datagrams received (ppd byte 5) |
| DBUGSW | | Debugging information |
| | 0 | state of system operation 0 = normal 1 = debugging 2 = standalone 3 = standalone fast startup |
| | 1 | state of BUGCHK handling 0 = proceed 1 = breakpoint |
| DEVCHR | (P1) | Device characteristics word, as described under the DVCHR JSYS in Chapter 3, except that B5 (DV%AV) is not meaningful. |
| DEVNAM | (P1) | SIXBIT device name including unit number, e.g., MTA3 |
| DEVUNT | (P1) | LH: Job number to which device is assigned (with ASND), or -1 if device is not assigned, or -2 if reserved for device allocator. RH: unit number, or -1 if device has no units (for example, DSK:) |
| DRMERR | | Information on drum errors |
| | 0 | number of recoverable errors |
| | 1 to n | varies depending on type of drum being used |
| DSKERR | | Information on disk errors |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|--------|--------|--|--|
| | 0 | | number of recoverable disk errors |
| | 1 to n | | varies depending on type of disk being used |
| DWNTIM | | | Downtime information |
| | 0 | | date and time when system will be shut down next |
| | 1 | | date and time when system will subsequently be up |
| HQLAV | | | High queue load averages |
| JBONT | Job # | | Owning job for CRJOB-created jobs. |
| JOBNAM | Job # | LH: reserved for DEC RH: index into the system program tables for the system program being used by this job (determined by the last SETSN call executed by the job) | |
| JOBPNM | Job # | | SIXBIT name of program running in this job |
| JOBRT | Job # | | CPU time used by the job (negative if no such job) |
| JOBTY | Job # | LH: controlling terminal line number, or -1 if none (job is detached) RH: reserved for Digital | |
| LOGDES | | | Logging information |
| | 0 | | designator for logging information |
| | 1 | | designator for job 0 and error information |
| LQLAV | | | Low queue load averages |
| MONVER | | | Monitor version number (contents of location 137) |
| NCPGS | | | One-word table containing number of pages of real (physical) user core available in system. Note that this value includes resident variables, and thus not all of the pages can be assigned to a user process. |
| NETRDY | | | ARPANET operational status table |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|--------|--|----------|--|
| 0 | | 0 | IMP down |
| | | .GT.0 | IMP going down |
| | | -1 | IMP up |
| 1 | | 0 | = network off, |
| | | non-zero | = network on |
| 2 | | | flags for NETSER (not for user) |
| 3 | | | time of last NCP cycle up |
| 4 | | | last IMP GOING DOWN message |
| | | B0-15 | reserved |
| | | B16-17 | 0 panic |
| | | | 1 scheduled hardware PM |
| | | | 2 software reload |
| | | | 3 emergency restart |
| | | B18-21 | number of 5-minute intervals before IMP goes down |
| | | B22-31 | number of 5-minute intervals IMP will be down |
| 5 | | | time of last IMP ready drop |
| 6 | | | time of last IMP ready up |
| 7 | | | time of IMP GOING DOWN message |
| NSWPGS | | | Default swapping pages |
| PTYPAR | | | Pseudo-TTY parameter information |
| | | 0 | LH: number of PTYS in system |
| | | | RH: TTY number of first PTY |
| QTIMES | | 0 to n | Accumulated runtime of jobs on the n scheduler queues |
| SCOUNT | | (P3) | Count of SETSN JSYSs for each subsystem |
| SNAMEs | | (P3) | SIXBIT name of system program, or 0 if this entry is unused in this and the corresponding four tables. |
| SNBLKS | | (P3) | Number of samples in working set size integral |
| SPFLTS | | (P3) | Total number of page faults of system program |
| SSIZE | | (P3) | Time integral of working set size |
| STIMES | | (P3) | Total runtime of system program |
| SYMTAB | | | SIXBIT table names of all GETAB tables |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

SYSTAT

Monitor statistics. The entries in this table are as follows:

| | |
|----|---|
| 0 | time with no runnable jobs |
| 1 | waiting time with 1 or more runnable jobs (waiting for page swapping) |
| 2 | time spent in scheduler |
| 3 | time spent processing pager traps |
| 4 | number of drum reads |
| 5 | number of drum writes |
| 6 | number of disk reads |
| 7 | number of disk writes |
| 10 | number of terminal wakeups |
| 11 | number of terminal interrupts |
| 12 | time integral of number of processes in the balance set |
| 13 | time integral of number of runnable processes |
| 14 | exponential 1-minute average of number of runnable processes |
| 15 | exponential 5-minute average of number of runnable processes |
| 16 | exponential 15-minute average of number of runnable processes |
| 17 | time integral of number of processes waiting for the disk |
| 20 | time integral of number of processes waiting for the drum |
| 21 | number of terminal input characters |
| 22 | number of terminal output characters |
| 23 | number of system core management cycles |
| 24 | time spent doing postpurging |
| 25 | number of forced balance set process removals |
| 26 | time integral of number of processes in swap wait |
| 27 | scheduler overhead time (same as entry 2) in high precision units |
| 30 | idle time (same as entry 0) in high precision units |
| 31 | lost time (same as entry 1) in high precision units |
| 32 | user time |
| 33 | time integral of number of processes on high queue. (High queue is high priority, low numerical value.) |
| 34 | time integral of number of processes on low queue. (Low queue is low priority, high numerical value.) |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | |
|----|--|
| 35 | sum of process disk-write waits |
| 36 | number of forced adjustments to balance set |
| 37 | integral of number of reserve pages of all processes in memory |
| 40 | integral of number of pages on replaceable queue. The replaceable queue contains pointers to all free memory pages. |
| 41 | high precision pager trap time |
| 42 | number of context switches |
| 43 | high precision time spent on background tasks. These tasks include low-level data transfer in communications layers, including network and terminal service routines. |
| 44 | total system page traps |
| 45 | total saves from replacement queue. A "save" occurs when a desired page is found on the replacement queue and need not be paged in. |
| 46 | number of pages removed from memory during system-wide garbage collection |
| 47 | integral of number of working sets in memory |
| 50 | wait time without swap waits in high precision units |
| 51 | count of working set loads |
| 52 | count of runnable processes removed from balance set |
| 53 | number of pages removed from memory during process-wide garbage collection |
| 54 | count of terminal input wakeups |
| 55 | count of read-after-write disk verifications |
| 56 | lowest,,highest active job on the system (does not include job 0) |
| 57 | operator,,user jobs logged into this system (does not include not logged in jobs) |

NOTE

This table is subject to
change (usually additions)
as measuring routines are
added to the system.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|--------|--------|---|
| SYSVER | | An ASCII string identifying the system name, version, and date. The string has the following format: string, TOPS-20 Monitor n.m(o)-p where "string" is the text contained in the file structure:<SYSTEM>MONNAM.TXT, "n" is the major version number (1 to 3 digits), "m" is the minor version number (0 to 2 digits), "o" is the edit number (1 to 6 digits), and "p" is the number of the group that last edited the version (0 or 1 digit). If "m" is zero, it and its preceding period are omitted. If "p" is zero, it and its preceding hyphen is omitted. Otherwise, the period and the hyphen are stored along with the other information, including the spaces and parentheses as shown, in the table. |
| TICKPS | | One-word table containing number of clock ticks per second. |
| TTYJOB | line # | LH: positive job number for which this is the controlling terminal, or -1 for unassigned line, or -2 for line currently being assigned, or job number to which this line is assigned. RH: -1 if no process is waiting for input from this terminal; other than -1 if some process is waiting for input. |
| WHOJOB | | Number of operator jobs and user jobs logged in (not including Job 0). |

The system program being run by a specific job may be determined from SNAMEs, using an index obtained from table JOBNAM.

The following monitor calls are used for obtaining information. Calls marked with an asterisk ("*") require privileges for specific functions.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | |
|---------|---|
| CNFIG% | Returns system configuration information |
| ERSTR | Translates an error number to a string |
| ESOUT | Returns an error string |
| GETAB | Returns a word from a system table |
| GETER | Returns the last error condition |
| GETJI | Returns job information for specified job |
| GETNM | Returns the program name being used by the job |
| GJINF | Returns job information for current job |
| GTAD | Returns the system's date |
| GTDAL | Returns the disk allocation of a directory |
| GTDIR* | Returns directory information |
| GTRPI | Returns the paging trap information |
| GTRPW | Returns the trap words |
| HPTIM | Returns the high-precision clock values |
| LATOP%* | Performs Local Area Transport (LAT) functions |
| MRECV* | Retrieves IPCF message |
| MSEND* | Sends IPCF message |
| MSTR* | Performs structure-related functions |
| MUTIL* | Performs IPCF functions |
| NTINF% | Returns generic network information |
| SKED* | Manipulates scheduler data base |
| SYSGT | Returns values for a system table |
| RUNTM | Returns the runtime of a job or process |
| TIME | Returns the time since the system was restarted |

2.4 COMMUNICATING WITH DEVICES

The monitor calls in this group are used to communicate with the devices on the system. Some of these devices are line printers, magnetic tapes, terminals, and card readers.

Many of the monitor calls in this group take a device designator as an argument. This designator can be either

```
LH: .DVDES(600000)+device type number
RH: unit number for devices that have units, arbitrary code for
    structures,
    or -1 for non-structure devices that do not have units
```

or

```
LH: 0
RH: .TTDES(400000)+terminal number, or .CTTRM(0,, -1) for
    controlling terminal
```

The STDEV monitor call is used to convert a string to its corresponding device designator.

The various devices are listed in the following table.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Table 2-3: Device Types

| Name | Description | Type | Symbol | Units |
|-------|--------------------------|------|--------|-------|
| DSK: | disk structure | 0 | .DVDSK | no |
| MTA: | magnetic tape | 2 | .DVMTA | yes |
| MT: | logical magnetic tape | 2 | .DVMTA | yes |
| LPT: | spooled line printer | 7 | - | yes |
| PLPT: | physical line printer | 7 | .DVLPT | yes |
| CDR: | spooled card reader | 10 | - | yes |
| PCDR: | physical card reader | 10 | .DVCDR | yes |
| FE: | front-end | | | |
| | pseudo-device | 11 | .DVFE | no |
| TTY: | terminal | 12 | .DVTTY | yes |
| PTY: | pseudo-terminal | 13 | .DVPTY | yes |
| NUL: | null device | 15 | .DVNUL | no |
| TCP: | ARPA network | 16 | .DVNET | no |
| CDP: | spooled card punch | 21 | - | yes |
| PCDP: | physical card punch | 21 | .DVCDP | yes |
| DCN: | DECnet active component | 22 | .DVDCN | no |
| SRV: | DECnet passive component | 23 | .DVSRV | no |

Device-designators may be formed for the devices shown above by taking the given symbolic device-type and adding .DVDES (600000).

The null device is an infinite sink for unwanted output and returns an EOF on input.

Device-dependent status bits are defined for some devices. These bits can be set or returned with the SDSTS or GDSTS call, respectively.

When an assignable device is assigned (by the ASND call) or opened (by the OPENF call) by one job, other jobs cannot do the following:

1. Assign the device with ASND.
2. Execute an OPENF call for the device, even if the JFN properly represents the device.

Structures are not restricted to these limitations; more than one user can simultaneously execute the OPENF call for files on structures.

There are some restrictions on the use of universal device designators and numeric designators in extended sections. Refer to Section 1.2.7.1 for this information.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The following sections describe many of the devices listed in the table above. The sections are in alphabetic order by generic device type (thus PCDR: and CDR: are listed under "c").

2.4.1 Physical Card Reader (PCDR:)

The following device-dependent status bits are defined for the card reader. These bits can be obtained with the .MORST function of the MTOPR call.

Table 2-4: PCDR: Status Bits

| Bit | Symbol | Meaning |
|-----|--------|---|
| B0 | MO%COL | Device is on line. |
| B10 | MO%FER | Fatal hardware error. This error generates an interrupt on software channel .ICDAE. (Refer to Section 2.6.1.) |
| B12 | MO%EOF | Card reader is at end of file. |
| B13 | MO%IOP | I/O in progress. |
| B14 | MO%SER | Software error. (Would generate an interrupt on an assignable channel.) |
| B15 | MO%HE | Hardware error. (Would generate an interrupt on software channel .ICDAE.) |
| B16 | MO%OL | Device is off line. |
| B17 | MO%FNX | Device is nonexistent. |
| B31 | MO%SFL | Output stacker full. |
| B32 | MO%HEM | Input hopper empty. |
| B33 | MO%SCK | Stack check. |
| B34 | MO%PCK | Pick check. |
| B35 | MO%RCK | Read check. |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.4.2 Spooled Card Reader (CDR:)

On most systems, the physical card reader devices (PCDR: devices) are under the control of the card reader spooler, SPRINT, and thus the ordinary user cannot open a PCDR: device, and must instead open a spooled card reader device (CDR:).

When a GTJFN is performed on device CDR:, the device characteristics (returned by DVCHR) are the same as those for device PCDR:. Thus, CDR: devices have units, and a unit number may be specified for the GTJFN.

When the OPENF is performed, However, the device characteristics become the same as device DSK:. This is because data read from device CDR: is actually read from a file in the spool directory <SPOOL>. The file is spooled from the PCDR: device to the spool directory by SPRINT.

Thus device CDR: is effectively a disk device, and no monitor call that can be used only to set the characteristics of a PCDR: device can be used for a CDR: device. Also, disk-only operations (such as PMAP) should not be done for a CDR: device. Both ASCII and image mode are supported for CDR: devices.

2.4.3 Physical Card Punch (PCDP:)

The following device-dependent bits are defined for the card reader. These functions can be obtained with the .MORST function of the MTOPR monitor call.

Table 2-5: PCDP: Status Bits

| Bit | Symbol | Meaning |
|-----|--------|--|
| B10 | MO%FER | Fatal error condition. |
| B12 | MO%EOF | All pending output has been processed. |
| B13 | MO%IOP | Output in progress. |
| B14 | MO%SER | Software error has occurred (would generate interrupt on an assignable channel). |
| B15 | MO%HE | Hardware error has occurred (would generate interrupt on channel .ICDAE). |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|-----|--------|--|
| B16 | MO%OL | Card-punch is off-line. This bit is set when operator intervention is required (card jam, hopper empty, stacker full). |
| B17 | MO%FNX | Card punch doesn't exist. |
| B32 | MO%HEM | Stacker is full or hopper is empty. |
| B33 | MO%SCK | Stacker is full or hopper is empty (same as above). |
| B34 | MO%PCK | Pick check. |

2.4.4 Spooled Card Punch (CDP:)

On most systems, the physical card punch devices (PCDP: devices) are under the control of the card punch spooler, SPROUT, and thus the ordinary user cannot open a PCDP: device, and must instead open a spooled card punch device (CDP:).

When a GTJFN is performed on device CDP:, the device characteristics (returned by DVCHR) are the same as those for device PCDP:. Thus, CDP: devices have units, and a unit number may be specified for the GTJFN.

However, when the OPENF is performed, the device characteristics become the same as device DSK:. This is because data written to device CDP: is actually written to a file in the spool directory <SPOOL>. The file is then spooled from the spool directory to the PCDR: device by SPROUT.

Thus device CDP: is effectively a disk device, and no monitor call that can be used only to set the characteristics of a PCDP: device can be used for a CDP: device. Also, disk-only operations (such as PMAP) should not be done for a CDP: device. Both ASCII and image mode are supported for CDP: devices.

2.4.5 Physical Line Printer (PLPT:)

The line printer normally accepts the 128 7-bit ASCII character codes (0-177 octal). However, by specifying a byte size of 8 when opening the printer, a program can transfer 8-bit bytes. Thus, the program can take advantage of printers that have more than 128 characters.

Each code sent usually causes a graphic to be printed. (Note that on a 64-character printer, lower case letters are represented as upper case.) However, the carriage control characters do not cause a graphic

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

to be printed; instead they cause specific actions to be taken. The actions taken are determined by the translation RAM and the Vertical Formatting Unit. These actions can be redefined by the installation, and the method by which they are redefined depends on the type of printer being used.

For the LP10 printer, which has a carriage control tape, the installation must change the tape to redefine the resulting actions.

For the LP05 and LP14 printers, which have a direct access Vertical Formatting Unit and a programmable translation RAM, the installation can redefine the resulting actions by:

1. Reprogramming the VFU by changing the VFU file with the MAKVFU program and reloading this file and the RAM.
2. Reprogramming the translation RAM by changing the RAM file with the MAKRAM program and reloading this file.

Refer to the LPINI and MTOPR monitor calls for the functions used in loading the VFU and RAM files.

The default actions taken on the carriage control characters, along with the default channels that determine these actions, are as follows:

Table 2-6: PLPT: Control Characters

| ASCII Character Code | Default Channel | Name | Default Action |
|----------------------------|--------------------|--------------|--|
| 11 | | Tab | No vertical motion. Skips to the beginning of every 8th column on the same line. |
| 12 | 8 | Line feed | Skips to column 1 on the next line. The last six lines of each page are skipped. |
| 13 | 7 | Vertical tab | Skips to column 1 on the line at the next third of a page. |
| 14 | 1 | Form feed | Skips to column 1 on the top of the next page. |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|----|---|-----------------|---|
| 15 | | Carriage return | No vertical motion. Returns to column 1 of the current line and does not advance the paper. |
| 20 | 2 | Half page | Skips to column 1 on the next half page. |
| 21 | 3 | Alternate lines | Skips to column 1 on the next even line. |
| 22 | 4 | Three lines | Skips to column 1 on the next of every third line. |
| 23 | 5 | Next line | Skips to column 1 on the next line without skipping the last six lines on a page. |
| 24 | 6 | Sixth page | Skips to column 1 on the next sixth of a page. |

The association between the ASCII code and the channel is determined by the RAM. The association between the channel and the default action is determined by the VFU. Therefore, a change in the VFU changes the association between the channel and the action, which causes the ASCII code to be associated with the new action.

2.4.5.1 PLPT: Status Bits - The following device-dependent status bits are defined for the line printer. These bits can be obtained with the .MORST function of the MTOPR call.

Table 2-7: PLPT: Status Bits

| Bit | Symbol | Meaning |
|-----|--------|--|
| B0 | MO%LCP | Lower case printer. |
| B10 | MO%FER | Fatal hardware error. This error generates an interrupt on software channel .ICDAE (refer to Section 2.6.1). |
| B12 | MO%EOF | All data sent to the printer has actually been printed. |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|-----|--------|--|
| B13 | MO%IOP | I/O in progress. |
| B14 | MO%SER | Software error (for example, interrupt character, page counter overflow). |
| B15 | MO%HE | Hardware error. Forms must be realigned. This error generates an interrupt on software channel .ICDAE. |
| B16 | MO%OL | Device is off line. |
| B17 | MO%FNX | Device is nonexistent. |
| B30 | MO%RPE | RAM parity error. |
| B31 | MO%LVU | Optical VFU. |
| B33 | MO%LVF | VFU error. |
| B34 | MO%LCI | Character interrupt. This generates an interrupt on channel .ICDAE. |
| B35 | MO%LPC | Page counter register overflow. |

2.4.6 Spooled Line Printer (LPT:)

On most systems, the physical line printer devices (PLPT: devices) are under the control of the line printer spooler, LPTSPL and thus the ordinary user cannot open a PLPT: device and must, instead, open a spooled line printer device (LPT:)

When a GTJFN is performed on device LPT:, the device characteristics (returned by DVCHR) are the same as those for device PLPT:. Thus, LPT: devices have units, and a unit number may be specified for the GTJFN. However, when the OPENF is performed, the device characteristics become the same as device DSK:. This is because data written to device LPT: is actually written to a file in the spool directory PS:<SPOOL>. When device LPT: is closed, the file in <SPOOL> is closed and a message sent to the line printer spooler LPTSPL causing it to print the file on the line printer.

Thus device LPT: is effectively a disk device, and none of the monitor calls that can be used only to set the characteristics of a PLPT: device can be used for a LPT: device. Also, disk-only operations (such as PMAP) should not be performed for LPT: devices. Note that LPTSPL writes only 7-bit bytes, so opening a LPT: device with any other byte size will cause erroneous results. Also, only ASCII mode is supported for LPT: devices.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.4.7 Physical Magnetic Tape (MTA:)

The following device-dependent bits are defined for magnetic tape.

Table 2-8: MTA: Status Bits

| Bit | Symbol | Meaning |
|-------|--------|---|
| 18 | MT%ILW | Drive is write protected |
| 19 | MT%DVE | Device error (hung or data late) |
| 20 | MT%DAE | Data error |
| 21 | MT%SER | Suppress automatic error recovery procedures |
| 22 | MT%EOF | Device EOF (file) mark |
| 23 | MT%IRL | Incorrect record length (not the same number of words as specified by the read operation or not a whole number of words) |
| 24 | MT%BOT | Beginning of tape |
| 25 | MT%EOT | End of tape |
| 26 | MT%EVP | Even parity |
| 29-31 | MT%CCT | Character counter if MT%IRL is on. In the case of an error generated by an incorrect record length, this field contains the number of bytes actually transferred. |
| 32 | MT%NSH | The selected data mode or density is not supported by the hardware (such as using ANSI-ASCII mode on a TMO3 controller). |

Data transfers to and from the magnetic tape can be performed using either buffered or unbuffered I/O.

2.4.7.1 Buffered I/O - The monitor uses buffered I/O when the sequential I/O calls (for example, BIN/BOUT, SIN/SOUT) are used to read from or write to the magnetic tape. When the tape is opened for sequential I/O (data mode .GSNRM on the OPENF call), the monitor

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

reserves buffer space large enough to hold two records of data. The maximum size of the records is specified with the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR monitor call. The maximum record lengths for magnetic tapes supported by TOPS-20 are listed in the description of the .MOSRS function of the MTOPR monitor call. The buffers reserved by the monitor allow the user's program to overlap computation with the transfer of data to and from the tape.

The BIN monitor call is used to read one byte from the tape, with the monitor filling one buffer with data as the user program is reading bytes from the other buffer. A program reading data from the tape with successive BIN calls obtains a stream of bytes until a tape mark is read. The SIN monitor call is used to read a specified number of bytes with the monitor again performing the double buffering. Both the BIN and the SIN calls read across record boundaries on the tape. The SINR monitor call is used to read variable-length records from the tape because each call returns one record to the user program. If the record on the tape contains more data than the SINR call requests, the remaining bytes in the record are discarded. The SINR call never reads across record boundaries on the tape. Thus, each SINR call begins reading at the first byte of the next record on the tape. With all three calls, the specified record size must be at least as large as the largest record being read from the tape.

The BOUT monitor call is used to write one byte on the tape. A program writing data on the tape with successive BOUT calls writes a stream of bytes packed into records of the specified size. The SOUT monitor call is used to write a specified number of bytes into one record equal to the given record size. The SOUTR call is used to write variable-length records on the tape because each call writes at least one record. The size of the record is equal to either the number of bytes specified in the SOUTR call or the number of bytes specified in the maximum record size, whichever is smaller. If the number of bytes requested in the call is greater than the specified record size, then records of the maximum size are written, plus another record containing the remaining bytes. If the end of tape marker is reached during sequential mode output, the data is written and an error return is given. Bit MT%EOT (bit 25) in the device status word will be set to indicate this condition.

When a CLOSF monitor call is executed for a magnetic tape to which buffered output is being done, any data remaining in the monitor's buffers will be written to the tape. The monitor writes two tape marks after the last record written and backspaces over the second mark. This allows a subsequent write operation to overwrite the last tape mark, and always leaves two tape marks (a logical end of tape) after the last record written.

The monitor does not write records of less than four words long. Thus if the user requests less than four words to be written on a SOUTR or DUMPO (see below) call, the monitor writes a four-word record,

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

completing it with zeros. On a SOUT call, if less than four words remain in the buffer at the time of the CLOSF call, the monitor again fills the record with zeros.

2.4.7.2 Unbuffered I/O - The DUMPI and DUMPO monitor calls are used to read from or write to the magnetic tape without using buffered I/O. (Unbuffered I/O is sometimes called dump mode I/O.) Unbuffered I/O uses a program-supplied command list to determine where to transfer data into or out of the program's address space. The command list can contain three types of entries:

1. IOWD n, loc transfers n words from loc through loc+n-1. The next command is obtained from the location following the IOWD. Each IOWD word reads or writes a separate magnetic tape record.
2. XWD 0, y takes the next command from location y.
3. 0 terminates the command list.

Refer to the DUMPI call description for more information.

On input, a new record is read for each IOWD entry in the command list. If the IOWD request does not equal the actual size of the record on the tape, an error (IOX5) is returned. The GDSTS monitor call can then be executed to examine the status bits set and to determine the number of bytes transferred. In addition, if a tape mark is read, an error (IOX4) is returned. On output, a new record is written for each IOWD entry in the command list.

There are two modes available in unbuffered I/O. In the normal mode, the monitor waits for the data transfer to complete before returning control to the program. In the no-wait mode, the monitor returns control immediately after queuing the first transfer so that the program can set up the second transfer. The monitor then waits for the first transfer to complete before queuing the second. If the first transfer is successful, the second one is started, and control is returned to the program. If the first transfer is not successful, an error is returned in AC1, and the second one is not started. The desired mode is specified by bit DM%NWT in AC1 on the DUMPI or DUMPO call.

2.4.7.3 Magnetic Tape Status - The status word of a magnetic tape can be obtained with the GDSTS call or individual status bits can be obtained with the MTOPR call. The GDSTS call waits for all activity to stop during sequential mode output, dump mode, and spacing operations before obtaining the status. A GDSTS call executed during sequential mode input returns the status of the current record.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Reading from or writing to a magnetic tape cannot be done if there are any errors set in the device status word. The program can clear errors with the SDSTS call or the .MOCLE function of the MTOPR call.

2.4.7.4 Reading a Tape in the Reverse Direction - With the .MOSDR function of the MTOPR call, the program can cause the tape to move in the reverse direction (toward the beginning of the tape) during read operations. The data in each record are returned in the forward order, but the records themselves are returned in the reverse order. The sensing-foil marking the beginning of tape is treated as an EOF tape mark.

When the SINR call is used to read data in the reverse direction, the byte size and record length specified in the call should equal the byte size and record length of the records on the tape. If the record characteristics specified in the call do not equal the characteristics of the records on tape, the bytes are returned out of phase with the bytes in the tape record.

When the SINR call is used to read data in the reverse direction, the number of bytes requested by the call should be at least as large as the size of the record on the tape. If the requested number is smaller than the number of bytes in the tape record, the remaining bytes in the record are discarded from the beginning of the record and not from the end of the record.

2.4.7.5 Hardware Data Modes - By using the .MOSDM function of the MTOPR call, the program can set the mode for storing data on a magnetic tape. The following descriptions indicate how bits are stored in the tracks and the number of frames required to store a 36-bit word of data.

The parity bit is represented in the diagrams by "P".

NOTE

Data undergoes 2 transformations before it is actually written to magnetic tape. The first transformation occurs when a word of data is formed into frames by the tape controller. The formats of these frames are illustrated in the diagrams below.

A second transformation occurs when the tape drive receives a frame of data from the controller, and physically writes that frame to tape: the bits within the frame are rearranged and then written. This final format is standardized throughout the computer industry and is designed to (among other things) place

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

the parity bit in the center of the tape (the "safest" part of the tape). Because this final format is standardized, it is "invisible" and does not affect user programs in any way.

Programmers who must deal with the problem of transferring data between DEC machines and the machines of other vendors need only concern themselves with the formats shown below. Thus, while it is technically incorrect to think of the diagrams below as showing the physical format of a word stored on magnetic tape, it is convenient to do so, and this simplification is made in this manual.

Unbuffered (Dump) Mode

This mode stores a word of data as a 36-bit byte in five frames of a 9-track tape. Note that the fifth frame is partially used. This mode is normally the default mode.

| TRACKS | | | | | | | | | FRAMES |
|--------|-----|-----|-----|-----|-----|-----|-----|---|--------|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | P | 1 |
| B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 | P | 2 |
| B16 | B17 | B18 | B19 | B20 | B21 | B22 | B23 | P | 3 |
| B24 | B25 | B26 | B27 | B28 | B29 | B30 | B31 | P | 4 |
| B32 | B33 | B34 | B35 | 0 | 1 | 0 | 0 | P | 5 |

Industry Compatible Mode

This mode stores a word of data as four 8-bit bytes in four frames of a 9-track tape. On a read operation, four frames of 8-bit bytes are read, left-justified, into a word. The remaining four bits of the word are 0, or are copies of the parity bits, depending on the hardware; these bits are not data. On a write operation, the leftmost four 8-bit bytes (bits 0 through 31) of the word are written in four frames on the tape. The rightmost four bits (bits 32 through 35) of the word are ignored and are not written on the tape. This mode is compatible with any machine that reads and writes 8-bit bytes.

| TRACKS | | | | | | | | | FRAMES |
|--------|-----|-----|-----|-----|-----|-----|-----|---|--------|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | P | 1 |
| B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 | P | 2 |
| B16 | B17 | B18 | B19 | B20 | B21 | B22 | B23 | P | 3 |
| B24 | B25 | B26 | B27 | B28 | B29 | B30 | B31 | P | 4 |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

ANSI ASCII Mode

This mode stores a word of data as five 7-bit bytes in five frames of a 9-track tape. On a read operation, five frames of 7-bit bytes are read, left-justified, into a word. The remaining bits (bits 35) of each frame are ORed together, and the result is placed in bit 35 of the word. On a write operation, the leftmost five 7-bit bytes of the word are written in five frames on the tape. Bit 35 of the word must be zero to conform to ANSI standards. It is written into the high-order bit of the fifth frame, and the remaining high-order bits of the first four frames are 0. This mode is useful when transferring ASCII data from TOPS-20 to machines that read 8-bit bytes. This mode is available on any 9-track drive connected to a TM02 or DX20 tape controller.

| TRACKS | | | | | | | | FRAMES | |
|--------|-----|-----|-----|-----|-----|-----|-----|--------|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| 0 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | P | 1 |
| 0 | B7 | B8 | B9 | B10 | B11 | B12 | B13 | P | 2 |
| 0 | B14 | B15 | B16 | B17 | B18 | B19 | B20 | P | 3 |
| 0 | B21 | B22 | B23 | B24 | B25 | B26 | B27 | P | 4 |
| B35 | B28 | B29 | B30 | B31 | B32 | B33 | B34 | P | 5 |

SIXBIT Mode

This mode stores a word of data as six 6-bit bytes in six frames of a 7-track tape. This mode is the only supported hardware mode for 7-track tapes.

| TRACKS | | | | | | FRAMES | |
|--------|-----|-----|-----|-----|-----|--------|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| B0 | B1 | B2 | B3 | B4 | B5 | P | 1 |
| B6 | B7 | B8 | B9 | B10 | B11 | P | 2 |
| B12 | B13 | B14 | B15 | B16 | B17 | P | 3 |
| B18 | B19 | B20 | B21 | B22 | B23 | P | 4 |
| B24 | B25 | B26 | B27 | B28 | B29 | P | 5 |
| B30 | B31 | B32 | B33 | B34 | B35 | P | 6 |

High Density Mode

In this mode, two 36-bit words are stored in 9 frames. High density mode is available on any 9-track drive connected to a DX20 controller.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| TRACKS | | | | | | | | | FRAMES |
|--------|-----|-----|-----|-----|-----|-----|-----|---|--------|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | P | 1 |
| B8 | B9 | B10 | B11 | B12 | B13 | B14 | B15 | P | 2 |
| B16 | B17 | B18 | B19 | B20 | B21 | B22 | B23 | P | 3 |
| B24 | B25 | B26 | B27 | B28 | B29 | B30 | B31 | P | 4 |
| B32 | B33 | B34 | B35 | B0 | B1 | B2 | B3 | P | 5 |
| B4 | B5 | B6 | B7 | B8 | B9 | B10 | B11 | P | 6 |
| B12 | B13 | B14 | B15 | B16 | B17 | B18 | B19 | P | 7 |
| B20 | B21 | B22 | B23 | B24 | B25 | B26 | B27 | P | 8 |
| B28 | B29 | B30 | B31 | B32 | B33 | B34 | B35 | P | 9 |

2.4.8 Logical Magnetic Tape (MT:)

Logical magnetic tape devices are used so that the system operator can fulfill a MOUNT request with any available tape drive that meets the requirements of the MOUNT request. The user never knows and need not know which physical drive (MTA:) is mapped to the logical drive (MT:).

Some JSYS functions available for MTA: devices are not available for MT: devices. Also, MT: devices are commonly used in a tape-labeled environment which causes further restrictions in the JSYS functions available for MT: devices. See the appropriate JSYSs for any restrictions that may apply.

2.4.9 Terminal (TTY:)

Most monitor calls in this group return an error if the device referenced is assigned to another job. However, a process with WHEEL capability enabled can reference a terminal assigned to another job (as controlling terminal or with ASND). The monitor calls pertaining to terminals have no effect, or return default-value information, when used with other devices.

The following status bits are defined for TTYS.

| <u>Bit</u> | <u>Symbol</u> | <u>Meaning</u> |
|------------|---------------|--|
| B35 | GD%PAR | The TTY will tolerate a parity bit. Any program producing binary output for a TTY should check this bit to determine if it should apply parity. If parity is to be applied, the TTY must be opened with an 8-bit bytesize; otherwise, a 7-bit bytesize must be used. |

DECnet NVTs will not accept a parity bit.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.4.9.1 JFN Mode Word - Each terminal in TOPS-20 is associated with a mode word. This word can be read with the RFMOD call and changed with the SFMOD and STPAR calls. The SFMOD call affects only the modes that are program-related: wakeup control, echo mode, and terminal data mode; thus a program can execute a SFMOD call without affecting previously-established device modes. The STPAR call, on the other hand, affects fields that describe device parameters (mechanical characteristics, page length and width, case conversion, and duplex control). Table 2-9 shows the format of the JFN mode word.

Table 2-9: JFN Mode Word

| Bit | Symbol | Changed by | Function |
|-------|--|------------|---|
| 0 | TT%OSP | SFMOD | output suppress control (1=ignore output; 0=allow output) |
| 1 | TT%MFF | STPAR | has mechanical form feed |
| 2 | TT%TAB | STPAR | has mechanical tab |
| 3 | TT%LCA | STPAR | has lower case |
| 4-10 | TT%LEN | STPAR | page length |
| 11-17 | TT%WID | STPAR | page width |
| 18-23 | TT%WAK | SFMOD | wakeup control on: B18: not used B19: ignore the other TT%WAK bits B20: formatting control character B21: non-formatting control character B22: punctuation character B23: alphanumeric character |
| 24 | TT%ECO | SFMOD | echos on |
| 25 | TT%ECM | STPAR | echo mode |
| 26 | TT%ALK | TLINK | accept links |
| 27 | TT%AAD | TLINK | accept advice |
| 28-29 | TT%DAM .TTBIN .TTASC .TTATO .TTATE | SFMOD | terminal data mode 00: no translation 01: translate both echo and output 10: translate output only 11: translate echo only |
| 30 | TT%UOC | STPAR | upper case output control 0: do not indicate 1: indicate by 'X |
| 31 | TT%LIC | STPAR | lower case input control 0: no conversion 1: convert lower to upper |
| 32-33 | TT%DUM .TTFDX | STPAR | duplex mode 00: Full duplex |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|----|--------|-------|---|
| | .TTHDX | | 10: Character half duplex |
| | .TTLDX | | 11: Line half duplex |
| | | | 01: Reserved for DEC |
| 34 | TT%PGM | STPAR | pause-on-command mode (1=enable pause-on-command mode, 0=disable pause-on-command mode.) |
| | | | This function enables/disables the TOPS-20 feature that allows a user to manually stop TTY output with ^S and resume it with ^Q. See MTOPR function .MOXOF for pause-at-end-of-page mode. |
| 35 | TT%CAR | | system carrier state; on if line is a dataset and the carrier is on. |

Bit 0 (TT%OSP) implements the CTRL/O function. If this bit is set, all program output directed to the terminal is discarded. When the bit is off, program output is buffered and sent as usual. The current contents of the output buffer are not cleared when this bit is set; clearing the buffer must be done explicitly (by means of the CFOBF call) if output is to be stopped immediately. Any input function clears this bit.

Bits 1, 2, and 3 (TT%MFF, TT%TAB, and TT%LCA) define several of the mechanical capabilities of the terminal and affect character handling on both input and output. Form feeds and tabs are simulated if the terminal does not have the required mechanical capability, or if simulation has been requested by the SFCOC call.

Bits 4-10 (TT%LEN) determine the number of line feeds necessary to simulate a formfeed, or the number of lines to fit on the display screen. A 0 value means the declared length of the page is indefinitely large.

Bits 11-17 (TT%WID) determine the point at which the output line must be continued on the next line by inserting a carriage return-line feed. If 0, no line folding occurs.

Bits 18-23 (TT%WAK) define the particular class of characters that, when input from the terminal, will wake up a waiting program. Refer to Section 2.4.9.3 for the definitions of the wakeup classes. Note that the class-wakeup scheme is maintained for compatibility with older programs. Newer programs should use the .MOSBM function of the MTOPR JSYS as it has more resolution and causes less system load.

Bit 24 (TT%ECO) defines if echos are to be given. If this bit is off, echoing is turned off. This is useful when the program is accepting a password or is simulating non-standard echoing procedures.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Bit 25 (TT%ECM) defines when the echo will occur. If this bit is off, the echo will occur when the program reads the character. That is, the echo occurs immediately if the program is waiting for input or is deferred if the program is not waiting for input. This is the standard echo mode which produces a correctly ordered typescript (i.e., program input and output appear in the order in which they occurred). If this bit is on, the echo occurs as soon as the character is typed. Note that this mode may cause editing to appear out of order on the typescript. This occurs because editing is performed as the program reads the character and not necessarily when the echo occurs.

Bits 28-29 (TT%DAM) define the terminal data mode. The four possible data modes are:

- 00 Binary (.TTBIN), 8-bit input and output. There is no format control or control group translation and no echoing. However, ^S and ^Q are still under control of TT%PGM.
- 01 ASCII (.TTASC), 7-bit input and output, plus parity on for control group output. There is format control as well as simulation and translation of control group for input (echo) and output according to the control words given on the SFCOC JSYS. This is the usual terminal data mode.
- 10 Disable the translation of echo (.TTATO). In all other respects, same as .TTASC.
- 11 Disable the translation of output (.TTATE). Obeys the CCOC word on input only. In all other respects, same as .TTASC.

The last two data modes allow the user to selectively disable the translation of control characters for input or output. When translation is disabled, control characters are always sent. Simulation of formatting control characters is still performed if requested by the control words of the RFCOC or SFCOC JSYS or if the device does not have the required mechanical capability. The translation typically results in some control characters being indicated by graphics instead of being sent as is. For example, disabling the translation of output characters is appropriate for some display terminals when the program must send untranslated control characters to control the display, but requires that the control characters typed by the user be indicated in the usual way.

Bit 30 (TT%UOC) specifies that upper case terminal output is to be indicated by 'X (single quote preceding character that is upper case) if TT%LCA is not set. This is primarily intended for terminals that are not capable of lower case output.

Bit 31 (TT%LIC) specifies that lower case terminal input is to be translated to upper case and that codes 175 and 176 are to be converted to code 33. This is useful for older terminals that send codes 175 or 176 in response to the ALT or ESC key.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Bits 32-33 (TT%DUM) define the three duplex modes presently available. Full duplex (.TTFDX) requires the system to generate the appropriate echo for each character typed in. Character half duplex (.TTHDX) assumes the terminal will internally echo each character typed but will require an additional echo for formatting characters such as carriage return. Line half duplex (.TTLDX) is similar to character half duplex but does not generate a line feed echo after a carriage return.

Bit 34 (TT%PGM) specifies the output mode. In display mode, the user can create a pause in the output while he reads material that would otherwise quickly disappear off the screen. The output is stopped with the CTRL/S character and started with the CTRL/Q character. Also, output automatically stops whenever a page, as defined by TT%LEN, has been output; output is resumed with CTRL/Q.

Bit 35 (TT%CAR) indicates the carrier state. If the line is a dataset, this bit is on if the carrier is on. If the line is not a dataset, this bit is undefined.

2.4.9.2 Control Character Output Control - Each terminal has two control character output control (CCOC) words. Each word consists of 2-bit bytes, one byte for each of the control characters (ASCII codes 0-37). The bytes are interpreted as follows:

- 00: ignore (send nothing)
- 01: indicate by ^X (where X is the character)
- 10: send character code
- 11: simulate format action

The RFCOC and SFCOC monitor calls read and manipulate the CCOC words. Table 2-10 lists the ASCII code for each character.

2.4.9.3 Character Set - The following information describes each character in the TOPS-20 character set that is pertinent to the monitor calls in this group. The wakeup class (refer to TT%WAK in Section 2.4.9.1) is abbreviated as follows:

- F formatting control character
- C non-formatting control character
- P punctuation character
- A alphanumeric character

Refer to Section 2.4.9.2 for the explanation of the control character output control (CCOC) words.

The following table lists the wakeup classes for the TOPS-20 character set (ASCII):

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Table 2-10: Wakeup Classes/CCOC Word Bits

| ASCII Code | Wakeup Class | CCOC Word(bits) | Character or Control Character |
|------------|--------------|-----------------|--------------------------------|
| 0 | C | 1(B0,1) | Ctrl/@ null,break |
| 1 | C | 1(B2,3) | Ctrl/A |
| 2 | C | 1(B4,5) | Ctrl/B |
| 3 | C | 1(B6,7) | Ctrl/C |
| 4 | C | 1(B8,9) | Ctrl/D |
| 5 | C | 1(B10,11) | Ctrl/E |
| 6 | C | 1(B12,13) | Ctrl/F |
| 7 | C | 1(B14,15) | Ctrl/G bell |
| 10 | F | 1(B16,17) | Ctrl/H backspace |
| 11 | P | 1(B18,19) | Ctrl/I horizontal tab |
| 12 | F | 1(B20,21) | Ctrl/J line feed |
| 13 | C | 1(B22,23) | Ctrl/K vertical tab |
| 14 | F | 1(B24,25) | Ctrl/L form feed |
| 15 | F | 1(B26,27) | Ctrl/M carriage return |
| 16 | C | 1(B28,29) | Ctrl/N |
| 17 | C | 1(B30,31) | Ctrl/O |
| 20 | C | 1(B32,33) | Ctrl/P |
| 21 | C | 1(B34,35) | Ctrl/Q |
| 22 | C | 2(B0,1) | Ctrl/R |
| 23 | C | 2(B2,3) | Ctrl/S |
| 24 | C | 2(B4,5) | Ctrl/T |
| 25 | C | 2(B6,7) | Ctrl/U |
| 26 | C | 2(B8,9) | Ctrl/V |
| 27 | C | 2(B10,11) | Ctrl/W |
| 30 | C | 2(B12,13) | Ctrl/X |
| 31 | C | 2(B14,15) | Ctrl/Y |
| 32 | C | 2(B16,17) | Ctrl/Z |
| 33 | All | 2(B18,19) | Escape (Altmode) |
| 34 | C | 2(B20,21) | Ctrl/Backslash |
| 35 | C | 2(B22,23) | Ctrl/Right Square Bracket |
| 36 | CxD | 2(B24,25) | Ctrl/Uparrow |
| 37 | F | 2(B26,27) | Ctrl/Backarrow |
| 40 | P | | Space |
| 41 | P | | ! |
| 42 | P | | " |
| 43 | P | | # |
| 44 | P | | \$ |
| 45 | P | | % |
| 46 | P | | & |
| 47 | P | | ' |
| 50 | P | | (|
| 51 | P | |) |
| 52 | P | | * |
| 53 | P | | + |
| 54 | P | | , |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|---------|-----|------------------------|
| 55 | P | - |
| 56 | P | . |
| 57 | P | / |
| 60-71 | A | 0-9 |
| 72 | P | : |
| 73 | P | ; |
| 74 | P | < |
| 75 | P | = |
| 76 | P | > |
| 77 | P | ? |
| 100 | P | @ |
| 101-132 | A | Upper Case Letters A-Z |
| 133 | P | [|
| 134 | P | \ |
| 135 | P |] |
| 136 | P | ^ |
| 137 | P | _ |
| 140 | P | Accent (Grave) |
| 141-172 | A | Lower Case Letters a-z |
| 173(1) | P | Left Brace |
| 174(1) | P | Vertical Bar |
| 175(1) | P | Right Brace |
| 176(1) | P | Tilde |
| 177 | All | Delete (Rubout) |

NOTE

1. Escape(33) and Delete(177) are considered to be in all wakeup classes.
2. If the terminal has B31(TT%LIC) on in the JFN mode word, codes 175 and 176 are converted to code 33 on input.
3. The class-wakeup scheme is maintained for compatibility with older programs. New programs should use the .MOSBM function of the MTOPR JSYS, as it has more resolution (it allows a 4-word character mask to specify individual wakeup characters) and causes less system load (low-level monitor I/O routines are subjected to fewer wakeups). Both the SFMOD JSYS and the .MOSBM function set the same mask; however, SFMOD computes wakeup classes from the mask while .MOSBM uses character-oriented wakeups.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.4.9.4 Terminal Characteristics Control - The various types of terminals have different characteristics for output processing, depending on their type and speed. The characteristics that can be associated with terminals are:

1. Mechanical form feed and tab
2. Lower case
3. Padding after carriage return
4. Padding after line feed
5. Padding after mechanical tab
6. Padding after mechanical form feed
7. Page width and length
8. Cursor commands

Instead of setting each of these parameters for his line, the user can specify a terminal type number, which causes the appropriate parameters to be set. Refer to the STTYP monitor call. The defined terminal types, along with their characteristics, are listed below.

Table 2-11: Terminal Characteristics

| Number | Terminal | Symbol | Characteristics |
|--------|--------------|--------|---|
| 0 | TTY model 33 | .TT33 | No mechanical form feed or tab, has upper case only, no padding after carriage return and line feed, padding after tab and form feed, page width 72, page length 66 |
| 1 | TTY model 35 | .TT35 | Has mechanical form feed and tab, has upper case only, no padding after carriage return and line feed, padding after tab and form feed, page width 72, page length 66 |
| 2 | TTY model 37 | .TT37 | No mechanical form feed or tab, lower case, no padding after carriage return and line feed, |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|-----|--------------|--------|--|
| | | | padding after tab and form feed, page width 72, page length 66 |
| 3 | TI/EXECUPORT | .TTEXE | No mechanical form feed or tab, lower case, padding after carriage return only page width 80, page length 66 |
| 4-7 | | | Reserved for customer |
| 8 | Default | .TTDEF | No mechanical form feed or tab, lower case, full padding, page width 72, page length 66 |
| 9 | Ideal | .TTIDL | Has mechanical form feed and tab, lower case, no padding, no specified width and length |
| 10 | VT05 | .TTV05 | No mechanical form feed, has mechanical tab, has upper case only, no padding after carriage return and tab, padding after line feed and form feed, page width 72, page length 20, has cursor commands |
| 11 | VT50 | .TTV50 | No mechanical form feed or tab, has upper case only, no padding, page width 80, page length 12, has cursor commands |
| 12 | LA30 | .TTL30 | No mechanical form feed or tab, has upper case only, full padding, page width 80, page length 66 |
| 13 | GT40 | .TTG40 | No mechanical form feed or tab, lower case, no padding, page width 80, page length 30 |
| 14 | LA36 | .TTL36 | No mechanical form feed or tab, lower case, no padding, page width 132, page length 66 |
| 15 | VT52 | .TTV52 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24 |
| 16 | VT100 | .TT100 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|----|--------------|--------|---|
| | | | When used in VT52 mode, the terminal type should be set to .TTV52. |
| 17 | LA38 | .TTL38 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 132, page length 66. |
| 18 | LA120 | .TT120 | Has mechanical form feed and tab, lower case, no padding, page width 132, page length 60 |
| 35 | VT125 | .TT125 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands and graphics capabilities |
| 36 | VK100 | .TTK10 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 84, page length 24, has cursor commands and color graphics capabilities |
| 37 | VT102 | .TT102 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands |
| 39 | VT131 | .TT131 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands |
| 40 | VT200 series | .TT200 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands; some models may have additional features |
| 52 | VT300 | .TT300 | No mechanical form feed, has mechanical tab, lower case, no padding, page width 80, page length 24, has cursor commands; some models may have additional features |

The STTYP monitor call sets the terminal type number for a line, and the GTTYP monitor call obtains the terminal type number.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.4.9.5 Terminal Linking - It is possible to link the output of any line to up to four other lines. The refuse/accept link bit TT%ALK (bit 26) in the JFN mode word controls terminal linking. If the bit is off for a particular terminal, a user cannot link to that terminal unless the user has WHEEL or OPERATOR privileges enabled. Although this bit can be read with the RFMOD monitor call, the bit can only be set with the TLINK call.

Refer to the TLINK monitor call for a description of terminal linking.

2.4.9.6 Terminal Advising - It is possible to receive advice from any terminal line in the system. The refuse/accept advice bit TT%AAD (bit 27) in the JFN mode word controls terminal advising. If this bit is off for a particular terminal, users cannot simulate typing on that terminal by means of the STI monitor call unless the user has WHEEL or OPERATOR privileges enabled. Although this bit can be read with the RFMOD monitor call, it can only be set with the TLINK call.

Refer to the TLINK monitor call for a description of terminal advising.

2.4.10 Transmission Control Protocol (TCP:)

The TCP: interface is consistent with other TOPS-20 network interfaces and uses standard TOPS-20 JSYSs for most functions. Any TCP: specific functions are accessible through the TCOPR% JSYS.

The programmer using the TCP: interface must provide information to the operating system about the virtual connection as well as various parameters dealing with the type and quality of service required. Connections are established using the GTJFN and OPENF JSYSs. Input/output is performed with the BIN, BOUT, SIN, SOUT, SINR, SOUTR, and TCOPR% calls. Status information is obtained from the TCOPR%, SOBF, and GDSTS calls.

2.4.10.1 GTJFN JSYS - The GTJFN JSYS is used to obtain an indexable file handle on a TCP: connection. The format of the GTJFN call for TCP: is the same as for any other GTJFN call (refer to GTJFN description in Chapter 3). The file name string for a TCP: GTJFN call specifies data about the desired connection.

The format for the GTJFN file specification is as follows:

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

TCP:[LOCALHOST-][LOCALPORT[#]].[FOREIGNHOST-][FOREIGNPORT][;A1...]

Bracket pairs indicate optional parameters.

"LOCALHOST-" specifies the local host address for this connection. This is useful for hosts that have multiple local addresses. The default for this field is the Internet address of the local host. This field can be specified using the alphanumeric host name or the octal host number. The "-" after the host name is required to delimit between the host name/number and the port number, and must be included even if the port number is omitted.

"LOCALPORT" specifies the local port number to use for this connection. The port number specified is in decimal. This field is optional. Port numbers are in the range 1 to 65535. Ports in the range 1 to 255 are special ports that require special privileges in order to be assigned. The "#" (must be preceded by a control-V) must be appended to a port in the range 1 to 255 and in the range 32768 to 65535 to prevent accidental assignment. Ports in the range 256 to 32767 are reserved for users. Ports in the range 32768 to 65535 are assigned as default port numbers on an as-needed basis. If no local port is specified, a number in the range 32768 to 65535 is assigned. A local port in the range 1 to 255 requires SC%WHL, SC%OPR, SC%NAS, or SC%NWZ privileges.

The "." is a delimiter separating the local host connection values from the foreign host connection values.

"FOREIGNHOST-" is used to specify the foreign host for which this connection is destined. It is an optional field. If the FOREIGNHOST is not specified, any host using any port is allowed to use this connection.

"FOREIGNPORT" is used to specify the foreign port for which this connection is destined. If this field is omitted, any foreign port is allowed to use this connection.

";A1..." specifies various attributes that this connection will use. The valid attributes are detailed in the GTJFN JSYS description in Chapter 3.

2.4.10.2 OPENF JSYS - The OPENF JSYS is used to force the TOPS-20 monitor to initiate the connection. The OPENF call is issued after a GTJFN call. Many parameters pertaining to this connection can also be set using the TCOPR% JSYS. Some parameters (for example, security levels) must be set before the OPENF JSYS is issued. The format of the OPENF call for TCP is the same as for other devices.

In TCP a connection is identified by both the foreign port and the local port. Two connections from system A to system B are allowed to

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

have the same local port or the same foreign port, but the connections cannot have both the same local and foreign ports simultaneously. A connection using a default local port will always be different from any other connection using a default local port. Wild connections (connections that allow any foreign host and/or foreign port) can be duplicated in multiple JFNs (in this job or other jobs).

2.4.10.3 Other JSYs - The SOUTR JSYS has the same functionality as the SOUT JSYS with one addition. The SOUTR JSYS sets the TCP PUSH flag for the last message generated by this call. This also forces all data currently held in local buffers to be sent immediately.

The SINR JSYS has the same functionality as the SIN JSYS with one addition. The SINR JSYS returns when a TCP message returns with the PUSH flag set. SINR also returns when the byte count is exhausted.

The following monitor calls are used for device control. Calls marked with an asterisk ("*") require privileges for specific functions.

| | |
|--------|--|
| ASND | Assigns a device |
| ATACH* | Attaches controlling terminal to a job |
| CFIBF | Clears terminal's input buffer |
| CFOBF | Clears terminal's output buffer |
| DEVST | Translates a device designator to a string |
| DIBE | Dismisses until terminal input buffer is empty |
| DOBE | Dismisses until terminal output buffer is empty |
| DTACH | Detaches controlling terminal from a job |
| DVCHR | Returns device characteristics |
| GDSKC | Returns disk usage |
| GDSTS | Returns the device status |
| GTTYP | Returns terminal type number |
| LPINI | Loads VFU or translation RAM |
| MSTR | Performs structure-dependent functions |
| MTOPR* | Performs device-dependent functions |
| MTU% | Performs functions for logical tape devices (MT: devices) |
| RELD | Releases a device |
| RELIQ% | Releases ownership of an Internet queue |
| RFCOC | Returns control character output control words |
| RFMOD | Returns the JFN mode word |
| RFPOS | Returns current position of the terminal |
| SDSTS | Sets the device status |
| SFCOC | Sets control character output control words |
| SFMOD | Sets program-related fields in the JFN mode word |
| SFPOS | Sets current position of the terminal |
| SIBE | Skips if input buffer is empty |
| SOBE | Skips if output buffer is empty |
| SOBF | Skips if output buffer is full |
| SPOOL | Defines and initializes input spooling |
| STDEV | Translates a string to a device designator |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

STPAR Sets device-related fields in the JFN mode word
STTYP Sets terminal type number
TLINK Controls terminal linking

2.5 SOFTWARE DATA MODES

I/O may be performed in one of several modes, depending on the device. (The mode is specified with the OPENF call.) The range of possible I/O modes is from 0 to 17 (octal). However, except for the TCP: device and less common hardware devices (such as paper-tape punches/readers) the only meaningful modes are 0, 10, and 17.

The following discussion lists the major devices supported by TOPS-20 and the applicable I/O modes:

| Device | Mode | Symbol | Explanation |
|--------|------|--------|---|
| CDP: | | | |
| PCDP: | 0 | .GSNRM | Normal mode - allows unit-record output. For card punches, this mode converts each 7-bit ASCII character to a 12-bit card-column code (Hollerith code) and outputs that code to the device. |
| | 10 | .GSIMG | Image mode - sends an "image" (rather than converting to Hollerith) of each byte. These are 12-bit bytes and are assumed to be in Hollerith code. If the device is opened with a byte size smaller than 12-bits, each byte sent is zero-padded on the left to form a 12-bit byte. |
| CDR: | | | |
| PCDR: | 0 | .GSNRM | Normal mode - allows unit-record input. For card readers, this mode converts each 12-bit card-column code (Hollerith code) to a 7-bit ASCII character and returns the ASCII character to the program. |
| | 10 | .GSIMG | Image mode - returns an "image" (rather than converting to ASCII) of the 12-bit card-code for each character read. In order to receive the full 12 bits, the program must use 12-bit bytes. |

Augmented image mode - this is a 16-bit version of image mode. The leftmost 4 bits are returned by the card reader controller. The first bit indicates that the column has a Hollerith error (and thus the card should

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

be rejected). The next 3 bits contain a value ranging from 0 to 7. If the value is from 1 to 7, it indicates that a punch occurred in that row. If the value is 0, it indicates that no punch occurred in columns 1 - 7. Effectively, a zero value indicates that a non-ASCII character was punched. This mechanism allows conversion to ASCII using a table with only 256 entries as opposed to a table with 4096 entries for 12-bit characters. This mode is available on PCDR: devices only and is used by specifying mode .GSIMG with a 16-bit bytesize.

| | | | |
|-------|----|--------|--|
| DCN: | | | |
| SRV: | 0 | .GSNRM | Normal mode - allows byte I/O. This device may be opened with 7, 8 or 36-bit bytes. However, all transfers are actually done with 8-bit bytes, and opening the device with an 8-bit bytesize will give the greatest efficiency. Requires DECnet software. .b.i-15 1 .GSSMB Small Buffer mode - allows small data segments to be transmitted to terminals. This mode is used by DECnet in communication with terminals, SRV:, and DCN: devices. |
| DSK: | 0 | .GSNRM | Normal mode - allows buffered byte, string, and paged I/O in 1 to 36-bit bytes. By definition, a DSK: device may be opened in any I/O mode, however the effect is the same as mode 0. |
| LPT: | | | |
| PLPT: | 0 | .GSNRM | Normal mode - allows buffered byte and string output. |
| PTY: | 0 | .GSNRM | Normal mode - for a PTY, the "mode" is merely used to open the device. The PTY will receive data according to the I/O mode of the TTY associated with it. |
| MTA: | | | |
| MT: | 0 | .GSNRM | Normal mode - allows buffered byte and string I/O. This is the most common I/O mode. |
| | 17 | .GSDMP | Dump mode - this mode is unbuffered by default (it can be set up for double-buffering) and is usually used to transfer blocks of data from tape to disk or disk to tape. For tape, a dump-mode read |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

(performed by DUMPI JSYS) performs reads on the basis of physical records. If less than a physical record is read, the data is transferred and an error is returned. A subsequent DUMPI will begin reading the tape at the start of the next physical record.

TCP: For TCP/IP systems only. All TCP data transfers are made with 8-bit bytes. 32-bit mode is only provided as an easy way to lower byte instructions required for data transfer. TCP: connections are full duplex. OF%RD and OR%WR must be set in the OPENF call.

- 0 .TCMWD Default mode - Same as .TCMWI.
- 1 .TCMWI Interactive mode. Wait for connection to be fully open before returning from the OPENF JSYS. Wait for the connection to be fully closed before returning from the CLOSF JSYS. Send all bytes as soon as possible (send data after each SOUT or BOUT). This mode attempts to give the most interactive response possible by sending many small messages. .ts 9,16,26
- 2 .TCMWH High throughput mode. Same action as mode zero for OPENF and CLOSF. Hold data in local buffers until accumulated bytes are sufficient for efficient transmission, or until transmission is requested with the TCOPR% or SOUTR JSYS. This mode attempts to give high throughput at low overhead by sending large messages.
- 3 .TCMII Immediate return mode. Return to user program immediately without waiting for a OPENF or CLOSF JSYS. Send all bytes as soon as possible (interactive mode).
- 4 .TCMIH Buffered immediate return mode. Same as mode 2 except that OPENF and CLOSF return immediately.

- NUL: 0 .GSNRM Normal mode
- 10 .GSIMG Image mode
- 17 .GSDMP Dump mode

The NUL device is a pseudo device used to "throw away" unwanted output from a program. The device may be opened in any mode.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | | |
|------|----|--------|---|
| TTY: | 0 | .GSNRM | Normal mode - allows buffered byte and string I/O. In this mode, format control and simulation and translation of control characters are performed by the monitor for input (echo) and output. (These services can be turned off by setting the appropriate bit in the JFN mode word.) Using an 8-bit bytesize in this mode implicitly changes the mode to .GSIMG (see below). |
| | 10 | .GSIMG | Image mode - allows buffered byte and string I/O, but disables format control and simulation and translation of control characters. On input, if the byte size is 8 bits, a parity bit (odd) is returned with the character. The parity bit is the high-order bit. On output, attempting to send an 8-bit byte that has incorrect parity may cause a device error. However, most terminals ignore a user-supplied parity bit. |

This mode can cause some reduction in the CPU time charged to a job for doing TTY output. The reduction is small, however, for TTY input. This is because the average process outputs many more characters than it inputs (the average ratio is approximately 20 characters output for each character input).

2.6 SOFTWARE INTERRUPT SYSTEM

The monitor calls in this group are used for controlling the software interrupt system. Note that if the program has an ERJMP or ERCAL after a monitor call that normally causes an interrupt on failure, the ERJMP or ERCAL overrides the interrupt. Refer to the TOPS-20 Monitor Calls User's Guide for an overview and description of the software interrupt system.

2.6.1 Software Interrupt Channels

Each interrupt is associated with one of 36 software interrupt channels below. The user program can assign channels 0-5 and 23-35 to various conditions, such as terminal interrupts, IPCF interrupts, ENQ/DEQ interrupts, PTY conditions, and terminal buffers becoming empty. The remaining channels are permanently assigned to certain error conditions. Any channel may be used for program-initiated interrupts (IIC call).

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Table 2-12: Software Interrupt Channels

| Channel | Symbol | Meaning |
|---------|--------|---|
| 0-5 | | Assignable by user program |
| 6 | .ICAOV | Arithmetic overflow (includes NODIV) |
| 7 | .ICFOV | Arithmetic floating point overflow (includes FXU) |
| 8 | | Reserved for DIGITAL |
| 9 | .ICPOV | Pushdown list (PDL) overflow[1] |
| 10 | .ICEOF | End of file condition |
| 11 | .ICDAE | Data error file condition[1] |
| 12 | .ICQTA | Disk full or quota exceeded when creating a new page[1] |
| 13-14 | | Reserved for DIGITAL |
| 15 | .ICILI | Illegal instruction[1] |
| 16 | .ICIRD | Illegal memory read[1] |
| 17 | .ICIWR | Illegal memory write[1] |
| 18 | | Reserved for DIGITAL |
| 19 | .ICIFT | Inferior process termination or forced freeze |
| 20 | .ICMSE | System resources exhausted[1] |
| 21 | | Reserved for DIGITAL |
| 22 | .ICNXP | Reference to non-existent page |
| 23-35 | | Assignable by user program |

[1] These channels are panic channels and cannot be completely deactivated. (Refer to Section 2.6.5.)

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.6.2 Software Interrupt Priority Levels

Each channel is assigned to one of three priority levels. The priority levels are numerically referenced as level 1, 2, or 3 with level 1 being the highest level interrupt. Level 0 is not a legal priority level. If an interrupt request occurs in a process where the level associated with the channel is 0, the system considers the process not prepared to handle the interrupt. The process is then frozen or terminated according to the setting of SC%FRZ (bit 17) in its capabilities word. (Refer to Section 2.7.1.)

2.6.3 Software Interrupt Tables

Before using the software interrupt system, a process must set up the following two tables and declare their addresses with the XSIR% or SIR calls.

LEVTAB

A 3-word table, indexed by priority level minus 1. There are two forms of this table.

In the general form, each word contains the 30-bit address of the first word of a two-word block in the process address space. The block addressed by word *n* of LEVTAB is used to store the global PC flags and address when an interrupt of level *n*+1 occurs.

The PC flags are stored in the first word of the PC block, and the PC address is stored in the second. This form of the table must be used with the XSIR% and XRIR% monitor calls, and can be used in any section.

The older form of the interrupt level table can be used in any single-section program, and must be used with the SIR and RIR calls. This table also contains three words, indexed by the priority level minus 1. Each word contains zero in the left half, and the 18-bit address of the word in which to store the one-word section-relative PC in the right half.

CHNTAB

A 36-word table, indexed by channel number. This table also has two formats.

The general format, for use with the XSIR% and XRIR% calls, can be used in any section of memory. Each word contains, in bits 0-5, the priority level (1, 2, or 3) to assign to interrupts generated on that channel; and in bits 6-35, the starting address of the routine to process interrupts generated on that channel.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

In the older format, for use with the SIR and RIR calls by any single-section program, the left half of each word contains the priority level (1, 2, or 3) for that channel. The right half contains the address of the interrupt routine that will handle interrupts on that channel.

2.6.4 Terminating Conditions

If an interrupt is received on a channel that is activated, but the interrupt cannot be initiated, then one of the following conditions exist:

1. The interrupt system for the process is not enabled (EIR JSYS) and the channel on which the interrupt occurred is a panic channel.
2. The table addresses have not been defined (SIR call).
3. No priority level has been assigned to the channel (i.e., left half of channel's word in CHNTAB is 0).
4. The channel has been "reserved" by the superior process (refer to the SIRCM call description).

This interrupt is considered a process termination condition. In this case the process that was to have received the interrupt is halted or frozen according to the setting of SC%FRZ (bit 17) in its capabilities word, and a process termination interrupt is sent to its superior. The superior process can then execute the RFSTS call to determine the status of the inferior process.

2.6.5 Panic Channels

Panic channels (refer to Section 2.6.1) cannot be completely deactivated by disabling the channel or the entire interrupt system. A software interrupt received on a panic channel that has been deactivated will be considered a process terminating condition. However, panic channels will respond normally to the channel on/off and read channel mask monitor calls.

2.6.6 Terminal Interrupts

There are 36 (decimal) codes used to specify terminal characters or conditions on which interrupts can be initiated. A process can assign a character or condition to any one of the program-assignable interrupt channels with the ATI call. Once the particular code is

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

assigned to a channel and the channel is activated (by means of AIC), occurrence of the character or condition corresponding to the code causes an interrupt to be generated. The terminal codes, along with their associated conditions, are shown in the table below.

Table 2-13: Terminal Interrupt Codes

| Terminal Code | Symbol Character or Condition |
|---------------|--|
| 0 | .TICBK CTRL/@ or break |
| 1 | .TICCA CTRL/A |
| 2 | .TICCB CTRL/B |
| 3 | .TICCC CTRL/C |
| 4 | .TICCD CTRL/D |
| 5 | .TICCE CTRL/E |
| 6 | .TICCF CTRL/F |
| 7 | .TICCG CTRL/G |
| 8 | .TICCH CTRL/H |
| 9 | .TICCI CTRL/I (tab) |
| 10 | .TIC CJ CTRL/J (line feed) |
| 11 | .TICCK CTRL/K (vertical tab) |
| 12 | .TICCL CTRL/L (form feed) |
| 13 | .TICCM CTRL/M (carriage return) |
| 14 | .TICCN CTRL/N |
| 15 | .TICCO CTRL/O |
| 16 | .TICCP CTRL/P |
| 17 | .TICCQ CTRL/Q |
| 18 | .TICCR CTRL/R |
| 19 | .TICCS CTRL/S |
| 20 | .TICCT CTRL/T |
| 21 | .TICCU CTRL/U |
| 22 | .TICCV CTRL/V |
| 23 | .TICCW CTRL/W |
| 24 | .TICCX CTRL/X |
| 25 | .TICCY CTRL/Y |
| 26 | .TIC CZ CTRL/Z |
| 27 | .TICES Escape (altmode) |
| 28 | .TICRB Delete (rubout) |
| 29 | .TICSP Space |
| 30 | .TICRF Dataset Carrier Off |
| 31 | .TICTI Typein |
| 32 | .TICTO Typeout |
| 33 | .TITCE Two-character escape sequence (see MTOPR%) |
| 34-35 | Reserved for Digital |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The terminal code `.TICRF` (30) is used to generate an interrupt when the dataset carrier state changes from on to off. Although any process can enable for this interrupt, only the top-level process in an attached job is guaranteed to receive it when the carrier state changes. However, a detached process is not guaranteed to receive this interrupt. If other processes enable for the interrupt, they can receive the interrupt either when the carrier state changes to off or later when the job is reattached after the detach caused by the carrier-off condition. In general, the occurrence of the change in the dataset carrier state is usable only by the top-level process.

The terminal codes `.TICTI` (31) and `.TICTO` (32) are used to generate interrupts on receipt of any character instead of a specific character. The `.TICTI` code generates an interrupt when the terminal's input buffer becomes nonempty (that is, when a character is typed and the buffer was empty before the input of the character). The `.TICTO` code generates an interrupt when the terminal's output buffer becomes nonempty. Note that neither one of these codes generates an interrupt if the buffer is not empty when the character is placed into it. The `SIBE` and `SOBE` calls can be used to determine if the buffers are empty.

The `.TITCE` code (33) generates an interrupt when the user types a special two-character escape sequence. It is set by the `.MOTCE` function of the `MTOPR JSYS`.

The frozen or unfrozen state (refer to Section 2.7.3.1) of a process determines if the interrupt is initiated immediately. Terminal interrupts are effectively deactivated when a process is frozen, even though the interrupts are indicated in the process' terminal interrupt word (obtained with the `RTIW JSYS`). When the process is unfrozen, the terminal interrupts are automatically reactivated.

When an operation is completed that explicitly changes the terminal interrupt word for the job (for example, a process freeze or unfreeze operation), the interrupt word for the job (and for the terminal line if the job is attached) is set to the inclusive OR (IOR) of all the unfrozen processes in the job. When an interrupt character is received, frozen processes are not considered when searching for a process to interrupt.

The user cannot directly access the actual terminal interrupt word. However, by specifying a process identifier of -5 as an argument to the `RTIW` or `STIW JSYSs`, he can read or change the terminal interrupt enable mask. The function of this mask is to allow processes to turn off interrupt codes activated by superior processes. Normally, the mask is -1, thereby enabling all terminal interrupts to be activated. A zero in any position of the mask prevents the corresponding terminal interrupt from being active. However, the fact that a code has been activated is remembered, and the code is activated when the mask is changed with a one in the corresponding position. Note that the process must have `SC%CTC` enabled in its capabilities word (refer to Section 2.7.1) to activate the terminal code for `CTRL/C` interrupts.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The SCTTY monitor call can be used to change the source of terminal interrupts for a process. Note that the process must have SC%SCT enabled in its capabilities word (refer to Section 2.7.1) to change the source of terminal interrupts.

2.6.6.1 Terminal Interrupt Modes - TOPS-20 handles the receipt of a terminal interrupt character in either immediate mode or deferred mode. An interrupt character handled in immediate mode causes the initiation of a software interrupt immediately upon its receipt by the system (as soon as the user types it). An interrupt character handled in deferred mode is placed in the input stream and initiates a software interrupt only when the program attempts to read it from the input buffer. In either case, the character is not passed to the program. If two occurrences of the same deferred interrupt character are received without any intervening character, the interrupt has an immediate effect. To detect this situation, the system maintains a separate one-character buffer in case the input buffer is otherwise full. The system assumes that interrupts are to be handled immediately unless the process has declared them deferred with the STIW monitor call.

The purpose of deferred mode is to allow interrupt actions to occur in sequence with other actions in the input stream. However, with multiple processes, the deferred interrupt occurs when any process of the job reads the interrupt character. If this process is the one enabled for the interrupt, it will be interrupted before any more characters are passed to the program. If the process to be interrupted is the top process, then the interrupt occurs before more characters are passed to the program, unless another process is also reading from the same source (usually an abnormal condition). If neither of the above situations applies, then the process doing terminal input continues to run and may receive several characters before the interrupt can take effect. This is unavoidable since the process doing input and the process to be interrupted are logically running in parallel.

2.6.7 Dismissing an Interrupt

Once the processing of an interrupt is complete, the user's interrupt routine returns control to the interrupted process by means of the DEBRK call. When the DEBRK call is executed, the monitor examines the contents of the return PC word to determine where to resume the process. If the PC word has not been changed, the process is restored to its state prior to the interrupt. For example, if the process was dismissed waiting for I/O to complete, it is restored to that state after execution of the DEBRK call. If the PC word has been changed, the process resumes execution at the new PC location.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The process can determine if an interrupt occurred during the execution of monitor code or user code by examining the user/exec mode bit (bit 5) of the return PC word. If the bit is on, the process was executing user code; if the bit is off, the process was executing monitor code (i.e., a JSYS). If the interrupt routine changes the return PC during the processing of an interrupt, the user-mode bit of the new PC word must be on. Note that the process may be executing monitor code but that the address portion of the PC is referencing a location in user code. To return to that user code location (i.e., to interrupt the execution of a monitor call), the process must turn on the user-mode bit.

The following monitor calls are used for controlling signals and synchronization. Calls marked with an asterisk ("*") require privileges for specific functions.

| | |
|--------|--|
| AIC | Activates interrupt channels |
| ATI | Assigns terminal code to channel |
| CIS | Clears the interrupt system |
| DEBRK | Dismisses current interrupt |
| DEQ* | Releases a resourcee locked by ENQ |
| DIC | Deactivates interrupt channels |
| DIR | Disables the interrupt system |
| DTI | Deassigns terminal code |
| EIR | Enables the interrupt system |
| ENQ* | Places a request in ENQ/DEQ resource queue |
| ENQC* | Returns status of a resource |
| GTRPI | Returns page trap information for specified process |
| GTRPW | Returns trap words |
| IIC | Initiates interrupts on specific channels in a process |
| MSTR* | Performs structure-related functions |
| MTOPR* | Performs device dependent functions |
| MUTIL* | Performs IPCF functions |
| NODE* | Performs DECnet functions |
| RCM | Reads activated channel word mask |
| RFSTS | Returns status of specified process |
| RIR | Reads the interrupt table addresses for a single-section program |
| RIRCM | Reads inferior reserved channel mask |
| RTIW | Reads terminal interrupt word |
| RWM | Reads waiting channel word mask |
| SCTTY | Changes source of terminal interrupts |
| SIR | Sets the interrupt table addresses for a single-section process |
| SIRCM | Sets inferior reserved channel mask |
| SKPIR | Skips if the interrupt system is enabled |
| STIW | Sets terminal interrupt word |
| SWTRP% | Intercepts arithmetic overflow or underflow conditions |
| TFORK* | Sets and removes monitor call intercepts |
| TIMER | Controls amount of time either a process within a job or the entire job can be run |
| XGTPW% | Returns page-fail words |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

XRIR% Reads the interrupt table addresses for a multiple-section program
XSIR% Sets the interrupt table addresses for a multiple-section process

2.7 PROCESS CAPABILITIES

The TOPS-20 system allows capabilities, such as the ability to examine the monitor and to enable for CTRL/C interrupts, to be given to certain processes. Each capability is separately protected and activated. The capabilities are assigned on a per-process basis, and their status is kept in the process' PSB.

The number of capabilities is limited to 36, and two words are used to store the status. For each capability, there is a bit in the first word that is set if the capability is available to the process. If the corresponding bit in the other word is also set, the capability is currently enabled. This allows the user to protect himself against accidental use without actually giving up the capability.

Inferior processes are created by superior processes (by means of the CFORK monitor call) with either no special capabilities or the capabilities of the creating process. Most capabilities relate to system functions and may be passed from superior to inferior process only if the superior itself has the capability. Some capabilities relate the inferior to the superior process, and may be given to an inferior whether or not available in the superior.

2.7.1 Assigned Capabilities

The following table lists the capabilities available for processes and jobs.

Table 2-14: Process/Job Capabilities

| Bit | Symbol | Meaning |
|-----------------------|--------|---|
| B0-8 Job Capabilities | | |
| 0 | SC%CTC | Process can enable for CTRL/C software interrupts. |
| 1 | SC%GTB | Process can examine monitor tables with the GETAB call. |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Note that the possession of this capability allows the process to do a GETAB. The capability need not be enabled.

3 SC%LOG Process can execute protected log functions (by means of the LGOUT JSYS).

Note that the possession of this capability allows the process to do a LGOUT. The capability need not be enabled.

6 SC%SCT Process can change the source of terminal interrupts for other processes.

B9-17 Capabilities that can be given to an inferior whether or not the superior itself has them. Of these, SC%FRZ (B17) cannot be changed by a process for itself.

9 SC%SUP Process can manipulate its superior process.

17 SC%FRZ Unprocessed software interrupts can cause the process to be frozen instead of terminated.

B18-35 User capabilities

18 SC%WHL User has wheel privileges.

19 SC%OPR User has operator privileges.

20 SC%CNF User has confidential information access.

21 SC%MNT User has maintenance privileges.

22 SC%IPC User has IPCF privileges.

23 SC%ENQ User has ENQ/DEQ privileges.

24 SC%NWZ User has ARPANET wizard privileges.

25 SC%NAS User has absolute ARPANET socket privileges.

26 SC%DNA User has access to DECnet.

27 SC%ANA User has access to ARPANET.

28 SC%SEO User has access to SEMI-OPERATOR.

User capabilities are originally established when the user's logged-in directory is created. (Refer to the CRDIR monitor call.)

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The capability word can be read with the RPCAP monitor call. Capabilities can be enabled with the EPCAP monitor call.

2.7.2 Access Control

It is often necessary for an installation to have more control over system resources than that offered by the process capability word. The following JSYSs allow each installation to write its own access-control program:

- o GETOK%
- o GIVOK%
- o RCVOK%
- o SMON
- o TMON

The access-control facility works as follows:

1. The installation writes its own access-control program. This program uses the SMON JSYS (privileged) to (1) enable or disable access checking for a variety of system resources and (2) allow or disallow access by default for those resources that are not explicitly checked by the access-control program.
2. The access-control program initializes itself and then issues the .SFSOK function of the SMON JSYS (privileged) to enable various types of access checking and to define itself as the access-control program.
3. The access-control program issues a RCVOK% JSYS (privileged). As the request queue is empty until a GETOK% request has been made, the RCVOK% JSYS causes the access-control program to block.
4. A system program or the monitor issues a GETOK% JSYS, causing an access request block to be appended to the GETOK% request queue (maintained by the monitor). The system program or monitor then blocks.
5. The monitor wakes up the access-control program and the blocked RCVOK% JSYS completes execution, retrieving the access request block from the GETOK% request queue. This block contains information supplied by the GETOK% call, plus certain job parameters.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

6. The access-control program determines whether to allow or deny the request and issues the GIVOK% JSYS (privileged) with the appropriate response for this request. The access-control program now issues another RCVOK% JSYS, which blocks or completes, depending on whether or not any additional requests are in the queue.
7. The system program or the monitor unblocks and gets a +1 return from the original GETOK% JSYS if the request has been granted, or gets an illegal instruction trap if the request has been denied.

Note the following characteristics of the access-control facility:

1. The GETOK% JSYS is imbedded in the code that is being protected against unauthorized use. For example, a DIGITAL-supplied GETOK% function allows access-control of the CRJOB JSYS; thus the TOPS-20 code that implements CRJOB will itself execute a GETOK% JSYS. An installation can also place GETOK% JSYSs in appropriate places in other software to provide additional access control.

However, this entire process is invisible to the ordinary user program. The only change such a program would encounter in an access-controlled environment would be the illegal instruction trap generated if the program attempted to use a protected resource that it was not entitled to use.

2. JSYSs performed by the access-control program or job 0 will not invoke access control.
3. After a system has been brought up, the first fork to execute the .SFSOK function of the SMON JSYS defines itself as the access-control fork. Any other fork that subsequently tries to issue a RCVOK% JSYS, a GIVOK% JSYS, or an SMON JSYS with function .SFSOK will receive an error.
4. The access-control facility has two timers associated with it:
 1. The time period between the execution of a GETOK% JSYS and its corresponding GIVOK% JSYS is measured. If the period exceeds a maximum, a BUGINF is generated on the CTY.
 2. The time period between the GETOK entry into the queue and the RCVOK% being executed is measured. If the period exceeds a maximum, a BUGCHK is generated on the CTY, all defaults are reestablished, the GETOK% request queue is flushed (the defaults are in effect for those requests also), and the monitor will no longer place GETOK% requests in the GETOK% queue.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.7.3 Processes and Scheduling

These monitor calls deal with establishing and interrogating the process structure of a job. Refer to the Monitor Calls User's Guide for an overview and description of the process structure.

2.7.3.1 Process Freezing - A superior process can cause one or all of its inferior processes to be frozen. A frozen process is one whose execution is suspended (as soon as it is stoppable from the system's point of view) in such a way that it can be continued at the point it was suspended. A process can be frozen directly or indirectly. A process is directly frozen when its superior makes an explicit request to freeze it. A process is indirectly frozen when its superior is frozen. When a process is directly frozen, all of its inferior processes are indirectly frozen. Therefore, a process can be both directly frozen by its superior process and indirectly frozen if its superior process is subsequently frozen.

The explicit unfreezing of a process clears both its direct freeze and the indirect freeze on all its inferior processes unless an inferior process has a direct freeze. The indirect unfreezing of a process clears only the freeze on that process. This means that an explicit freeze of a process prevents the running of any of its inferior processes, and an explicit unfreezing of a process automatically resumes its inferiors.

The FFORK and RFORC monitor calls are used to freeze and unfreeze processes, respectively. An argument of -4 to these calls directly freezes or resumes all immediately inferior processes, and any processes below the immediately inferior ones are indirectly frozen or resumed. (The freeze and unfreeze operations are never legal on any process that is not inferior to the one executing the monitor call.)

The frozen or unfrozen state of a process can only be changed directly. Thus, monitor calls like SFORK and HFORK change other states of a process but do not affect the frozen state. If the process is frozen and a call is executed that changes one of its states, the process remains frozen and does not begin operating in the changed state until it is resumed. For example, a program can change a frozen process's PC with the SFORK call, but the process will not begin running at the new PC until it is unfrozen. Similarly, the HFORK call can be executed on a frozen process, but the process will not be in the halted state until it is unfrozen. The changed status is always reflected in the information returned by the RFSTS call. In the first example above, RFSTS would return the changed PC, and in the second, it would return the halted code in the status word.

The following monitor calls are associated with capabilities and processes. Calls marked with an asterisk ("*") require privileges for specific functions.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | |
|--------|---|
| ADBRK | Controls address breaks |
| CFORK | Creates inferior process |
| CRJOB* | Creates a new job |
| DEQ* | Releases a resource locked by ENQ |
| DISMS | Dismisses process for specified amount of time |
| ENQ* | Places a request in the ENQ/DEQ resource queue |
| ENQC* | Returns status of a resource |
| EPCAP | Enables process capabilities word |
| FFORK | Freezes one or more processes |
| GETER | Returns last error condition for a process |
| GETNM | Returns program name currently in use by job |
| GFRKH | Gets process handle |
| GFRKS | Gets current process structure |
| GTRPI | Returns page trap information for a specified process |
| HALTF* | Halts a process |
| HFORK | Halts an inferior process |
| KFORK | Kills one or more processes |
| LGOUT* | Logs a job out |
| PRARG | Sets or returns process argument block |
| RESET | Resets and initializes current process |
| RFACS | Returns process' accumulators |
| RFORK | Resumes one or more processes |
| RFRKH | Releases process handles |
| RFSTS | Returns process' status |
| RMAP | Obtains a handle on a page in a process |
| RPACS | Returns accessibility of page |
| RPCAP | Returns process capabilities word |
| RSMAP% | Returns information about the mapping of one section of a process |
| RTFRK | Returns the handle of the process suspended because of a monitor call intercept |
| RWSET | Releases working set |
| SETJB* | Sets job parameters |
| SETER | Sets the last error condition encountered by process |
| SETNM | Sets private name of program in use by job |
| SETSN | Sets system name or private name of program in use by job |
| SFACS | Sets process' accumulators |
| SFORK | Starts a process in section zero |
| SPACS | Sets accessibility of page |
| SPLFK | Splices a process structure |
| TFORK* | Sets and removes monitor call intercepts |
| UTFRK | Resumes a process suspended because of a monitor call intercept |
| WAIT | Dismisses process until interrupt occurs |
| WFORK | Waits for process to terminate |
| WSMGR% | Manages working set of a process |
| XRMAP% | Extended read mapping |
| XSFRK% | Starts a process in a non-zero section |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

2.7.3.2 Execute-Only Files and Execute-Only Processes - The basic definition of an execute-only file is one that cannot be copied, read, or manipulated in the usual manner, but can be run as a program. An execute-only file has the following characteristics:

1. The file must be protected with execute access allowed, but with read access not allowed.
2. The file cannot be read or written using any of the file-oriented monitor calls (SIN, SOUT, BIN, BOUT, PMAP, for example).
3. The file can be mapped into a process (using GET), but only in its entirety and only into a virgin process. A process so created is called an execute-only process.

NOTE

A virgin process is one that has just been created (using CFORK). Furthermore, if a process is virgin, no operations have been performed on the process. This means no changes have been made to its address space, PC, ACs, interrupt system, or traps, and the process has not been mapped to a file or another address space.

4. Only disk-resident files can be considered execute-only.
5. A process with WHEEL or OPERATOR capabilities enabled can gain read access to any file and can thus circumvent the execute-only features of an execute-only file.

An execute-only process has the following characteristics:

1. An execute-only process can be started only at its entry vector.
2. A process that is created by an execute-only process and shares the same address space becomes execute-only itself.
3. No other process can read from an execute-only process' address space or accumulators.
4. No other process can change any part of an execute-only process' context in such a way as to cause the execute-only process to unintentionally reveal any part of its address space.
5. An execute-only process can not be prevented from mapping pages of its own address space into an inferior process. It is the programmer's responsibility to avoid revealing an execute-only process through its inferior forks.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

6. No JSYS explicitly indicates that a given process is execute-only. However, the RFACS JSYS will always fail for an execute-only process and can be used to determine this information, if it is required.

A program is execute-only for particular users based on its file protection. If a user tries to run a file and cannot read it, but does have execute access, a process is created as usual. The file is mapped into this virgin process, circumventing the read protection on the file. This process is then an execute-only process.

Users may select a file to be execute-only by allowing execute but not read access to the file. This can be done by setting the protection field for the desired class of users (owner, group, or world) to FP%EX+FP%DIR, or 12 octal. For example, to make a file execute-only for everybody except the owner of the file, the user would set the protection to 771212 octal.

The following JSYSs do not work for execute-only programs:

1. ADBRK - referring to an execute-only process
2. GET - referring to an execute-only process
3. PMAP - with either source or destination an execute-only process
4. SCVEC - referring to an execute-only process
5. SDVEC - referring to an execute-only process
6. SEVEC - referring to an execute-only process
7. SMAP% - with either source or destination an execute-only process
8. SPACS - referring to an execute-only process
9. XGVEC% - referring to an execute-only process
10. XSVEC% - referring to an execute-only process

The START command cannot be used with a start address argument for an execute-only process. A program that is execute-only must be written to protect itself. The program should not map itself out to inferior processes unless the entire address space is mapped. The program should not do a GET and execute programs in its address space over which it has no control.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Some programs cannot be made execute-only. Some major examples are:

- o Any object-time system, such as LIBOL or FOROTS. They must be merged into the address space and thus violate the restriction of reading an execute-only file into a virgin address space. Note that an execute-only process can merge in an object-time system, however.
- o The TOPS-10 compatibility package (PA1050). This has the same restriction that object-time systems have.
- o Any program that uses the TOPS-10 RUN or GETSEG UUOs. These UUOs require mapping into a non-virgin address space.
- o Any program that needs to be started at any location other than its entry vector (START or REENTER address).

2.8 SAVE FILES

A save file is a method of storing an executable memory image on disk. TOPS-20 handles two formats of save files: nonsharable (primarily intended for compatibility with TOPS-10) and sharable.

Save files use data compression to reduce the size of the on-disk copy. Non-sharable save files use word-oriented compression: memory words containing zero are not stored in the disk file. Sharable save files use page-oriented compression: memory pages in which all words contain zero are not stored in the disk file.

Shareable save files are generated with the TOPS-20 SAVE command or the SSAVE JSYS. Non-sharable save files are generated with the TOPS-20 CSAVE command or the SAVE JSYS. The formats of the two types of save files are discussed below.

2.8.1 Format for Nonsharable Save Files

The format of a nonsharable save file is as follows:

```
IOWD      length, address at which to put "length" data words
"length" data words
IOWD      length, address at which to put "length" data words
"length" data words
.
.
.
```

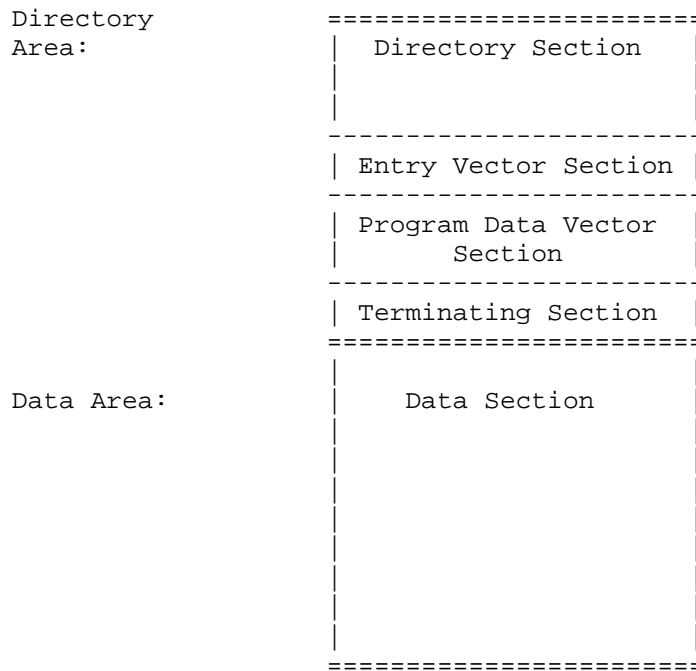
FUNCTIONAL ORGANIZATION OF MONITOR CALLS

XWD length of entry vector, pointer to first word of entry vector

2.8.2 Format of Sharable Save Files

A sharable save file is divided into two main areas: the directory area contains information about the structure of the file, and the data area contains the data of the file.

The following diagram illustrates the general format of a sharable save file:



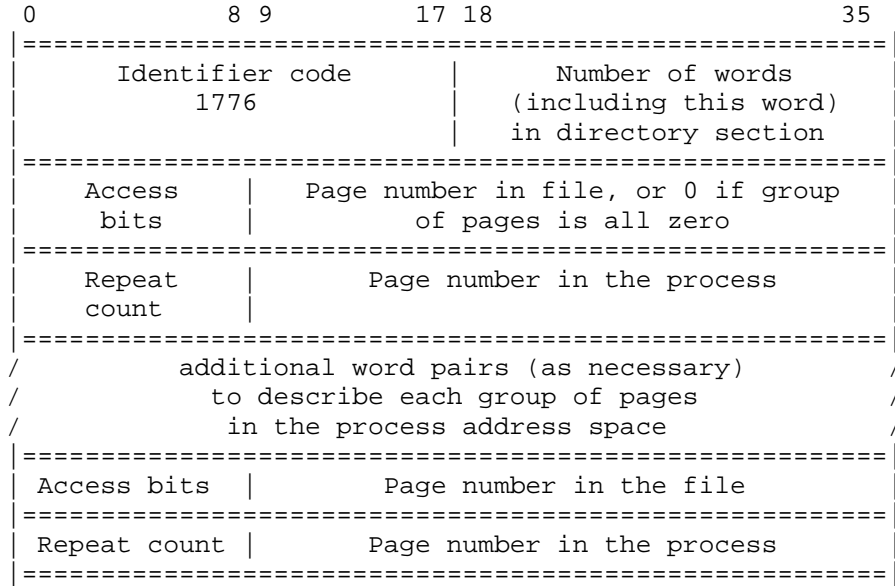
The directory area of the save file has four sections: the directory section, the entry vector section, the program data vector section, and the terminating section. The directory area may be from 1 to 3 pages long, depending on the access-characteristics of the pages in the data area of the save file. Although SSAVE% creates a directory area that is only one page long, there is no limit to the size of a directory area created with the SAVE% monitor call.

Each of the four sections in the directory area begins with a word containing its identifier code in the left half and its length in the right half. Each section is described in the paragraphs below.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The directory section is the first of the three sections and describes groups of contiguous pages that have identical access. The length of this section varies according to the number of groups that can be generated from the data portion of the save file. The more data pages that can be combined into a single group, the fewer groups required, and the smaller the directory section.

The format of the directory section is as follows:



The access bits are determined from the access bits specified by the user on the SSAVE monitor call. The bits currently defined in the directory section are:

- B1 The process pages in this group are sharable
- B2 The process pages in this group are writable

The remaining access bits in the directory section are zero.

The repeat count is the number (minus 1) of consecutive pages in the group described by the word pair. Pages are considered to be in a group when the following three conditions are met:

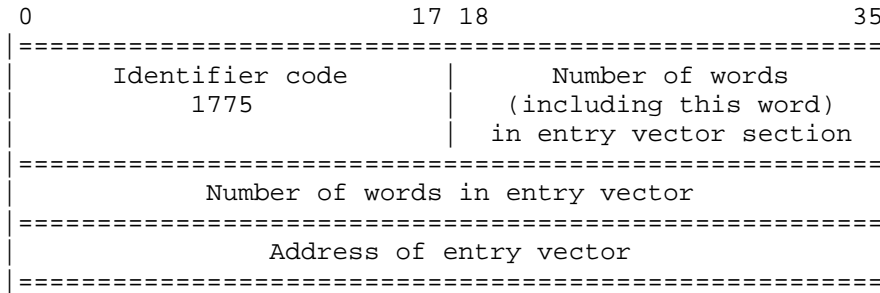
1. The pages are contiguous.
2. The pages have the same access.
3. The pages either are all zero or are all existent and readable.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

A page is considered to be all zero if it is nonexistent or is not readable. A page containing all zeros is considered to be existent. A group of all zero pages is indicated by a file page number of 0.

The word pairs are repeated for each group of pages in the address space.

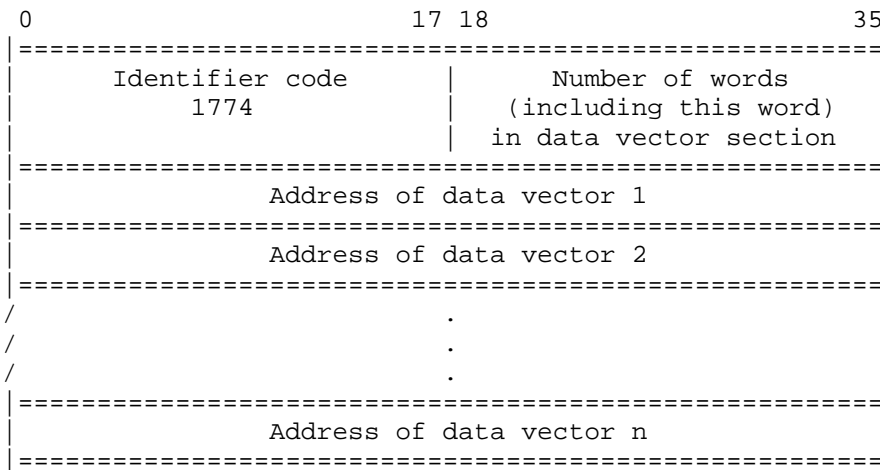
The entry vector section follows the directory section, and points to the entry vector. The format of the entry vector section is as follows:



This section contains the address of the entry vector. Refer to Section 2.8.3 for a description of the entry vector.

The program data vector section follows the entry vector section. The program data vector section contains the addresses at which the program data vectors begin (PDVAs). This section is optional, and only appears if the program declares some program data vectors.

The format of the program data vector section is as follows:



FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The terminating section follows the program data vector section. Its format is as follows:

| | |
|-----------------|---|
| Identifier code | |
| 1777 | 1 |

The remaining words in the last page of the save file are filled with zeros and are ignored by the monitor.

2.8.3 Entry Vector

The entry vector is a block of data that describes entry conditions to be used when the program in the process is executed. The first word of the entry vector contains the program start instruction, the second word contains the program reenter instruction, and the third word contains the program version number. (The version number format is: B0-B2(VI%WHO) containing the group who last modified the program, B3-B11(VI%MAJ) containing major version number, B12-B17(VI%MIN) containing minor version number, and B19-B35(VI%EDN) containing edit number. If B18(VI%DEC) is set, the version number fields are printed in decimal by the TOPS-20 command processor). Subsequent words in the entry vector can contain data applicable to the particular entry (refer to the GCVEC and GDVEC monitor calls).

Typically, the entry vector looks like this:

```
JRST start-addr
JRST reenter-addr
version number
.
.
.
```

Each process has an entry vector word in its process storage block. The format of the entry vector word is:

```
LH: length of the entry vector (1-777)
RH: address of the first word of the entry vector.
```

The data for this word is obtained from the entry vector in the save file when a GET monitor call is executed for the file.

Note that if the left half of the entry vector (usually the length) is 254000 (octal), then there is no real entry vector. The program start address is in the right half of location 120, the reenter address is in the right half of location 124, and the program version is in

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

location 137. This format is not recommended, but is maintained for compatibility with older monitors.

2.8.4 Program Data Vector

The program data vector (PDV) is a block of data that LINK writes into memory when loading and linking a program. The PDV resides in memory as a part of the program, and starts at a program data vector address (PDVA). User programs can use this data. Although TOPS-20 currently does not use the data in the PDV, words 13, 14, and 15 of the PDV are provided for possible future system use.

The format of the program data vector is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .PVCNT | Length of the PDV (including this word). |
| 1 | .PVNAM | Name of the program for which this data vector exists. The name is word-aligned ASCII, which means that the characters in the name are represented by seven-bit bytes, and that the first byte in each word begins with bit zero. |
| 2 | .PVEXP | Address of the exported information vector. |
| 3 | .PVREE | Reserved for DIGITAL. |
| 4 | .PVVER | Program version number. |
| 5 | .PVMEM | Address of a block of memory that contains data describing the program memory (a memory map). See the LINK manual, Appendix G, for a description of this block. |
| 6 | .PVSYM | Address of the program symbol table. |
| 7 | .PVCTM | Time at which the program was compiled. |
| 10 | .PVCVR | Version number of the compiler. |
| 11 | .PVLTM | Time at which the program was loaded. |
| 12 | .PVLVR | Version number of LINK. |
| 13 | .PVMON | Address of a monitor data block. (Not currently used.) |
| 14 | .PVPRG | Address of a program data block. (Not currently used.) |
| 15 | .PVCST | Address of a customer-defined data block. |

The PDVOP% monitor call manipulates PDVs. When loading a program into memory, LINK executes a PDVOP% call to give the monitor the addresses of the PDVs for that program. The PDVAs are the only data regarding PDVs that the monitor keeps in its data base.

Once the monitor knows the PDVAs for a program, other programs and other processes can use PDVOP% to obtain those PDVAs from the monitor. An inquiring program or process must use the PDVA (and another PDVOP% call) to obtain the data in the PDV.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The PDVOP% call also allows you to add PDVAs to, or delete PDVAs from, the monitor's data base. Refer to Chapter 3 for a complete description of PDVOP%.

The following monitor calls are used in conjunction with save files. Calls marked with an asterisk ("*") require privileges for specific functions.

| | |
|--------|--|
| GCVEC | Gets compatibility package entry vector |
| GDVEC | Gets RMS entry vector |
| GET* | Obtains a saved file |
| GEVEC | Gets process entry vector of a single-section program |
| PDVOP% | Obtains information about execute-only programs |
| SAVE | Saves a process as nonsharable |
| SCVEC | Sets compatibility package entry vector |
| SDVEC | Sets RMS entry vector |
| SEVEC | Sets the entry vector for a single-section program |
| SFRKV | Starts process using its entry vector |
| SSAVE | Saves a process as sharable |
| XGSEV% | Gets extended special entry vector |
| XGVEC% | Gets process entry vector for a multiple-section program |
| XSFRK% | Starts a process using a user-supplied, global PC |
| XSSEV% | Sets extended special entry vector |
| XSVEC% | Sets the entry vector for a multiple-section program |

2.9 INPUT/OUTPUT CONVERSION

The monitor calls in this group perform input/output conversion. Calls are available to convert in both directions between ASCII text (in core or in a file) and integer numbers, floating point numbers, and TOPS-20 internal dates and times.

2.9.1 Floating Output Format Control

2.9.1.1 Free Format - The most common format control used with the FLOUT JSYS is free format. This is specified by setting B18-23 (FL%FST) of the format control word to 0. (Refer to Section 2.9.1.2.) Normally, the entire format control word is set to 0; however, certain fields may be specified to force a particular output.

Most numbers greater than or equal to 10^{-4} but less than 10^6 (with some exceptions) are output in a typical FORTRAN F format. If the number is an exact integer, it is output with no terminating decimal point unless B6(FL%PNT) is on. If the number is a fraction, it is output as .xxxx with no leading zeros. Nonsignificant trailing zeros in the fraction are never output. A maximum of seven digits is output if the second field (FL%SND) is not specified. The sign of the number is output only if negative.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

If the number is outside the range above, it is output in a typical FORTRAN E format (with some exceptions). The exponent is output as Esxx, where s is the sign output only on negative exponents and xx are the digits of the exponent. The above exceptions about outputting the decimal point and suppressing trailing, nonsignificant zeros apply.

Another free format similar to that above is invoked by specifying a nonzero value for B13-17 (FL%RND) of the format control word. The value in this field specifies the place at which rounding should occur. If this value is 7, the output is the same as if the value were 0 as above. If this value is less than 7, rounding occurs at the specified place, but the output will be as above with a maximum of 7 digits (for example, 12360 with a rounding specification of 3 will output as 12400). If this value is greater than 7, rounding occurs at the specified position, but more than 7 digits are output. In this case, digits are output until either the rounding specification number is reached or until trailing, nonsignificant zeros are reached.

2.9.1.2 General Format Control - The format control word specifies the format for floating point output when free format is not desired. The control word indicates the desired output for the three fields of the number, plus additional control for items such as rounding. The first field of the number is up to the decimal point. The second field is from the decimal point to the exponent. The third field is the exponent.

The format control word is as follows:

Table 2-15: Floating-Point Format Control

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0-1 | FL%SGN | Sign control for first field. The first character position is always used for the minus for negative numbers. For positive numbers, the first character position is defined according to the values below: |
| | Value | Symbol Meaning |
| | 0 | .FLDIG First character is digit. |
| | 1 | .FLSPC First character is space. |
| | 2 | .FLPLS First character is plus sign. |
| | 3 | .FLSPA First character is space. |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|------|--------|---|
| 2-3 | FL%JUS | Justification control for first field. |
| | Value | Symbol Meaning |
| | 0 | .FLLSP Right justify number using leading spaces. |
| | 1 | .FLLZR Right justify number using leading zeros. |
| | 2 | .FLLAS Right justify number using leading asterisks. |
| | 3 | .FLTSP Left justify number up to decimal point using trailing spaces after third field. |
| 4 | FL%ONE | Output at least one digit (0 if necessary) in first field. |
| 5 | FL%DOL | Prefix the number with a dollar sign (\$). |
| 6 | FL%PNT | Output a decimal point. |
| 7-8 | FL%EXP | Third (exponent) field control. |
| | Value | Symbol Meaning |
| | 0 | .FLEXN No exponent field. |
| | 1 | .FLEXE Output E as first character of exponent field. |
| | 2 | .FLEXD Output D as first character of exponent field. |
| | 3 | .FLEXM Output *10^ as first characters of exponent field. |
| 9-10 | FL%ESG | Exponent sign control. The first character position is always used for the minus for negative exponents. For positive exponents, the first character position is defined according to the values below: |
| | Value | Symbol Meaning |
| | 0 | .FLDGE First character after exponent prefix is digit. |
| | 1 | .FLPLE First character after prefix is plus sign. |
| | 2 | .FLSPE First character after prefix is space. |
| | 3 | .FLDGT First character after exponent prefix is digit. |
| 11 | FL%OVL | Use free format on overflow of the first field and expand exponent on overflow of the third |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | | |
|-------|--------|--|
| | | field. If this bit is not set, no additional output occurs on column overflow. |
| 13-17 | FL%RND | Digit position at which rounding will occur. If field is 0, rounding occurs at the 12th digit. If field is 37, no rounding occurs. |
| 18-23 | FL%FST | Number of characters in first field, including a dollar sign (\$) if FL%DOL is set. (refer to FL%JUS). |
| 24-29 | FL%SND | Number of characters in second field. |
| 30-35 | FL%THD | Number of characters in third field. |

As an example, to output a number in the format xx.yy, the following bits should be set in AC3 of the FLOUT monitor call.

| | |
|-----------------|--|
| B4(FL%ONE) | output at least one digit in the first field |
| B6(FL%PNT) | output a decimal point |
| B13-B17(FL%RND) | do not round the number |
| B22 | output a maximum of two digits in the first field |
| B28 | output a maximum of two digits in the second field |

Examples of numbers output in this format are:

43.86 4.24 0.43

2.9.2 Date And Time Conversion Monitor Calls

TOPS-20 internal date and time is maintained in a 36-bit word and is based on Greenwich Mean Time. The date is in the left half and is the number of days since November 17, 1858; the time is in the right half and is represented as a fraction of a day. This allows the 36-bit value to be in units of days with a binary point between the left and right halves. The resolution is approximately one-third of a second; that is, the least significant bit represents approximately one-third of a second. The date changes at the transition from 11:59:59 PM to 12:00:00 midnight.

For conversions between local and internal date and time, the time zone in which the installation is located is normally used, with daylight savings applied from 2AM on the last Sunday in April to 1:59:59AM on the last Sunday in October.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Two monitor calls in this group, IDTIM and ODTIM, convert date and time between text strings (in core or in a file) and internal format. These should satisfy most users. However, there are four more calls, which are subsets of IDTIM and ODTIM. The calls ODTNC, IDTNC, ODCNV, and IDCNV make available separately the conversion between internal format date and time and separate numbers for local year, month, and day, and the conversion between those numbers and text strings. They also provide additional options, which give the caller more control over the conversion performed than IDTIM and ODTIM.

Time zones occur in the calling sequences of the latter four JSYSs. A time zone is represented internally as a number between -12 and 12 decimal, representing the number of hours west of Greenwich. For example, EST is zone 5. Zones -12 and 12 represent the same time but different days because the zones are on opposite sides of the international date line.

The following are examples of valid dates and times:

```
6-FEB-76
FEB-6-76
FEB 6 76
FEB 6, 1976
6 FEB 76
6/2/1976
2/6/76
```

Below are examples of valid times:

```
1:12:13
1234
16:30          (4:30PM)
1630
1234:56
1:56AM
1:56-EST
1200NOON
12:00:00AM     (midnight)
11:59:59AM-EST (late morning)
12:00:01AM     (early morning)
```

"AM" or "PM" can follow a time specification that is not greater than 12:59:59. "NOON" or "MIDNIGHT" can follow 12:00:00.

Any time specification can be followed by a dash and a time zone. Table 2-16 lists the time zones defined within TOPS-20, their abbreviations, and the left half of the word generated or accepted by the calls that read, write, or convert dates and times. The right half of the word ordinarily contains the time expressed as seconds after midnight.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

Table 2-16: Time Zones

| Zone Name | Abbreviation | Left half |
|-----------------------------|--------------|-----------|
| GREENWICH DAYLIGHT TIME | GDT | 700000 |
| GREENWICH MEAN TIME | GMT | 500000 |
| GREENWICH STANDARD TIME | GST | 500000 |
| ATLANTIC DAYLIGHT TIME | ADT | 700004 |
| ATLANTIC STANDARD TIME | AST | 500004 |
| EASTERN DAYLIGHT TIME | EDT | 700005 |
| EASTERN STANDARD TIME | EST | 500005 |
| CENTRAL DAYLIGHT TIME | CDT | 700006 |
| CENTRAL STANDARD TIME | CST | 500006 |
| MOUNTAIN DAYLIGHT TIME | MDT | 700007 |
| MOUNTAIN STANDARD TIME | MST | 500007 |
| PACIFIC DAYLIGHT TIME | PDT | 700010 |
| PACIFIC STANDARD TIME | PST | 500010 |
| YUKON DAYLIGHT TIME | YDT | 700011 |
| YUKON STANDARD TIME | YST | 500011 |
| ALASKA-HAWAII DAYLIGHT TIME | HDT | 700012 |
| ALASKA-HAWAII STANDARD TIME | HST | 500012 |
| BERING DAYLIGHT TIME | BDT | 700013 |
| BERING STANDARD TIME | BST | 500013 |
| LOCAL DAYLIGHT TIME | DAYLIGHT | 600000 |

All strings (for example, months, time zones, AM-PM-NOON-MIDNIGHT) can be represented by any nonambiguous abbreviation (for example, D-DECEMBER, M-MIDNIGHT).

Spaces are ignored before and between fields whenever they do not terminate the input string. This means spaces are not allowed before colons, AM,PM,NOON, and MIDNIGHT, the dash before the time zone, or the time zone. A tab is also allowed between the date and time.

The input string can be terminated by any nonalphanumeric character.

Monitor calls relating to date and time are as follows:

- IDTIM Inputs date and time, converting to internal format
- ODTIM Outputs date and time, converting from internal format to text
- IDTNC Inputs date and time without converting to internal format
- ODTNC Outputs date and time in internal format
- IDCNV Converts from day, month, year to internal date and time
- ODCNV Converts from internal date and time to day, month, year

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

GTAD Gets current date and time in internal format

2.10 ARCHIVE/VIRTUAL DISK SYSTEM

The following section defines terms that are used in the description of the archive/virtual disk system:

Virtual disk A storage technique in which the contents of some files reside on disk, while the contents of other files may reside on tape. When a file is "migrated" to tape, a copy of its FDB is left on disk and the file is deleted from disk. Note that the term "migration" applies only to files transferred to tape by the virtual disk system.

Archived file A file of unchanging data stored on magnetic tape. Although copies of the file may exist on disk, the original is stored on magnetic tape. When a file gains archive status, it can no longer be changed. If a writeable copy is desired, the COPY command must be used.

When a file is archived, the file contents are usually deleted from disk, leaving only the FDB on disk. However, it is possible to override the deletion process.

Offline/online A file is said to be offline if the file has been moved to tape by either the virtual disk system or the archive system. A file is said to be online if the original or a copy of it is on disk. A file may be offline, online, or both. A file that is offline and not online will have only its FDB stored on disk. In the last case, the FDB will contain pointers to the saveset and tape file number. This provides a link between the FDB on disk and the file on tape.

Invisible/visible An invisible file is one that does not appear in a simple DIRECTORY listing, and is not accessible to programs (unless the GTJFN specifically sets bit G1%IIN) and EXEC commands. A visible file appears in a DIRECTORY listing and is accessible to programs and EXEC commands.

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

The concept of an invisible file is primarily designed to make offline-only files transparent to the user. However, the invisible/visible status of a file may be changed regardless of whether the file is online, offline, archived, not archived, migrated, or not migrated.

The virtual disk system is designed to conserve disk space by moving selected files from disk to tape. Files are marked for migration to tape by the REAPER program. At the option of the system administrator, REAPER may mark files in any of the following three categories:

1. Files that have not been referenced within a specified period of time.
2. Online copies of migrated or archived files that have not been referenced within a specified period of time.
3. Files in a directory that is over permanent disk quota. If the directory contains a file named MIGRATION.ORDER, then REAPER uses that file as an order list for marking files. Otherwise REAPER follows the order given in the REAPER command list. Two REAPER passes are made with the first pass using the order specified in MIGRATION.ORDER or the REAPER command string. If the first pass fails to bring the directory under quota, the second pass will consider any file in the directory for migration.

The actual migration of disk files to tape is performed by a special DUMPER run. The actual run will occur periodically, with the length of the period determined by the system administrator.

File archiving is designed to write unalterable "permanent" copies of disk files on tape. The user voluntarily marks a file for archiving, and the next archive/virtual disk DUMPER run will archive the file.

For added security two tape copies of each archived or migrated file are made.

The following monitor calls are used to implement the archive/virtual disk system. Calls marked with an asterisk ("*") require privileges for specific functions.

| | |
|--------|---|
| ARCF* | Performs archive/virtual disk operations |
| CRDIR* | Creates or modifies a directory |
| DELDF* | Expunges deleted files |
| DELNF | Retains specified number of generations of file |
| GNJFN | Assigns a JFN to the next file |
| GTJFN | Assigns a JFN to a file |
| JFNS | Translates a JFN to a string |

FUNCTIONAL ORGANIZATION OF MONITOR CALLS

| | |
|--------|-----------------------------|
| OPENF | Opens a file |
| RFTAD | Reads file's time and dates |
| SETJB* | Sets job parameters |
| SFTAD* | Sets file's time and dates |
| TMON | Reads monitor flags |

2.11 PRIVILEGED MONITOR CALLS

The following monitor calls are privileged and require the process to have WHEEL or OPERATOR capability enabled.

| | |
|--------|--|
| ALLOC | Allocates a device to a particular job |
| ASNIQ% | Assigns TCP/IP Internet queue |
| ASNSQ | Assigns TCP/IP special message queue |
| BOOT | Performs functions required for loading front-end software |
| DIAG | Reserves and releases hardware channels |
| DSKAS | Assigns specific disk addresses |
| DSKOP | Allows hardware address specification in disk transfers |
| GIVOK% | Allows/denies access to a protected system resource |
| HSYS | Halts the monitor |
| IPOPR% | Performs Internet network management operations |
| LLMOP% | Low level maintenance operations |
| LPINI | Loads line-printer VFU |
| MDDT% | Enters MDDT program |
| MSFRK | Starts a process in monitor mode |
| MTALN | Associates magnetic tape drive with logical unit number |
| MTU% | Performs MT-device functions |
| NI% | TOPS-20 user interface to the Ethernet |
| NTMAN% | Performs DECnet network management functions |
| PEEK | Reads monitor data |
| PLOCK | Locks physical pages |
| PMCTL | Controls physical memory |
| RCVOK% | Services GETOK% requests |
| SCS% | Interface to System Communications Services |
| SJPRI | Sets job priority |
| SMON | Sets monitor flags |
| SNOOP | Performs system performance analysis |
| SPOOL | Performs spooling-related functions |
| SPRIW | Sets process priority |
| STAD | Sets system date/time |
| SYERR | Places information in the System Error file |
| TCOPR% | Internet terminal control operations |
| USAGE | Makes entries in accounting file |
| USRIO | Places program in user I/O mode |
| UTEST | Monitors executed instructions |
| XPEEK% | Monitor data retrieval functions |

The capabilities for a process are enabled by the EPCAP JSYS.

CHAPTER 3

TOPS-20 MONITOR CALLS

Gives a particular type of access to a given directory. The possible types of accesses are:

1. Connecting to a directory on a given structure.
2. Gaining owner and group access rights to directories on a structure without actually connecting to a directory on that structure.
3. Relinquishing owner and group access rights to directories on a structure without disconnecting from a directory on that structure.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled. Some functions require WHEEL or OPERATOR capability enabled.

When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: B0(AC%CON) Connect the job to the specified directory. After successful completion of the call, the job is connected to and has owner access to the directory. The job's default directory becomes this directory.

B1(AC%OWN) Give the job owner access to the specified directory and group access to directories in the same groups as the specified directory. The job's connected directory is unchanged. This function cannot be given for another job or for a files-only directory.

TOPS-20 MONITOR CALLS
(ACCES)

B2(AC%REM) Relinquish owner access (obtained with the AC%OWN function) to the specified directory and group access to directories in the same group. The job's connected directory is unchanged. This function cannot be given for another job or for a files-only directory. The settings of B0 and B1 are ignored if B2 is on and the job number given is for the current job.

B3(AC%PWD) Validate password by encrypting user-supplied password before doing compare.

B18-35 Length of the argument block

AC2: Address of the argument block

RETURNS +1: Always

Access cannot be given to a regulated structure unless the MSTR JSYS has been first used to increment the mount count. All structures are regulated by default except the primary structure or any structure that has been made nonregulated with the MSTR JSYS. Access rights and all JFNs on the regulated structure must be released before the mount count can be decremented.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .ACDIR | Byte pointer to ASCIZ string containing the structure and directory name or a 36-bit directory number. The ASCIZ string must be of the form structure:<directory>. |
| 1 | .ACPSW | Byte pointer to ASCIZ string containing the password of the specified directory. The password is not required if: <ol style="list-style-type: none"> 1. The directory is on a domestic structure and has the same name as the user's logged-in directory. 2. Function AC%CON is being done and the directory does not require a password for connecting. |
| 2 | .ACJOB | Number (decimal) of job or -1 for the current job. The process must have WHEEL or OPERATOR capability enabled to give a specific job number other than its own. |

TOPS-20 MONITOR CALLS
(ACCES)

The ACCES monitor call can be given for another job if the type of access being requested is for connecting the job (AC%CON) and if the process executing the call has WHEEL or OPERATOR capability enabled.

The ACCES monitor call is used to implement the CONNECT, ACCESS, and END-ACCESS commands of the TOPS-20 Command Language.

Generates an illegal instruction interrupt on error conditions below.

ACCES ERROR MNEMONICS:

ACESX1: Argument block too small
ACESX3: Password is required
ACESX4: Function not allowed for another job
ACESX5: No function specified for ACCES
ACESX6: Directory is not accessed
ACESX7: Directory is "files-only" and cannot be accessed
CNDIX1: Invalid password
CNDIX5: Job is not logged in
STRX01: Structure is not mounted
STRX02: Insufficient system resources
STRX03: No such directory name
STRX04: Ambiguous directory specification
STRX09: Prior structure mount required
STRX10: Structure is offline
LGINX2: Directory is "files-only" and cannot be logged into
CAPX1: WHEEL or OPERATOR capability required
RCDIX2: Invalid directory specification
ARGX07: Invalid job number
ARGX08: No such job

Controls address breaks. An address break is the suspension of a process when a specified location is referenced in a given manner.

ACCEPTS IN AC1: Function code in the left half and process handle in the right half

AC2: Function-specific argument

AC3: Function-specific argument

RETURNS +1: Always

This JSYS is useful when debugging a program. For example, consider the problem of debugging a program consisting of a fork running

TOPS-20 MONITOR CALLS
(ADBRK)

several inferior forks mapped to the same address space. One (or more) of the inferior forks is erroneously referencing a particular address. To find out which fork(s) are referencing that address, do the following:

1. Set up the software interrupt system for interrupts on channel 19.
2. Perform the ADBRK .ABSET function for each inferior process, using the handle of the inferior process and the address being erroneously referenced.
3. When a channel 19 interrupt occurs, perform an RFSTS JSYS for each inferior process. The interrupted process that caused the address break will have a code 7 (.RFABK) returned in its status word.
4. Perform the ADBRK .ABGAD function for each process that caused an address break. This returns the address of the instruction that erroneously referenced the break address.
5. Perform the RFORK JSYS to restart the process(es) halted by address break(s).
6. Continue running the program and repeating the last three steps until the program completes execution, or it no longer generates address breaks.

The ADBRK JSYS can also be used to find which instruction in a process references a wrong memory location. The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--------------------------------------|
| 0 | .ABSET | Set address break. |
| 1 | .ABRED | Read address break. |
| 2 | .ABCLR | Clear address break. |
| 3 | .ABGAD | Return address of break instruction. |
| 4 | .ABSRG | Set address break range. |
| 5 | .ABRRG | Read address break range. |
| 6 | .ABGBR | Return address break data. |

Each function is described in the following paragraphs.

TOPS-20 MONITOR CALLS
(ADBRK)

Setting address breaks - .ABSET

This function initializes the address break facility for the specified process. When the process references the location in the manner for which the break has been set, it is suspended. Its superior receives a software interrupt on channel 19 (.ICIFT) if it has enabled for that channel. After processing the interrupt, the superior process can resume the inferior by executing the RFORK monitor call.

Only one address break can be in effect for a process at any one time, and the break affects only the process for which it is set. If another process references the location on which a break is set, it is not affected by the break. When an address break is set in a page shared among processes and each process is to be suspended when it references the location, the ADBRK call must be executed for each process.

Breaks cannot be specified for the accumulators.

The .ABSET function requires the following arguments to be given:

AC2: address of location on which to break.

AC3: flag word indicating the type of reference on which to break. The following flags are currently defined:

B0(AB%RED) Break on a read reference.

B1(AB%WRT) Break on a write reference.

B2(AB%XCT) Break on an execute (instruction fetch) reference.

Reading address breaks - .ABRED

This function returns the current address break information for the specified process. It returns the following information on a successful return:

AC2: address of location on which a break is set

AC3: flag word indicating the type of reference on which the break will occur. The following flags are currently defined:

B0(AB%RED) Break will occur on a read reference.

B1(AB%WRT) Break will occur on a write reference.

B2(AB%XCT) Break will occur on an execute (instruction fetch) reference.

TOPS-20 MONITOR CALLS
(ADBRK)

If no address break has been set for the process, the contents of AC2 and AC3 are zero on return.

Clearing address breaks - .ABCLR

This function removes any address break that was set for the specified process. A program can also remove a break by executing the .ABSET function with AC2 and AC3 containing zero.

Returning the address of the break instruction - .ABGAD

This function returns in AC2 the address of the location on which the process encountered a break. When the location on which the break occurred is in a JSYS routine, the address returned is a monitor PC, not the address of the JSYS. The program can obtain the address of the JSYS by executing an RFSTS monitor call.

Setting an address break range - .ABSRG

This function is the same as .ABSET except it allows for the setting of a range of addresses on which to break. Currently the range is restricted to a single address location. This function requires that AC2 contain the address of an argument block. The format of the argument block is:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .ABHDR | Flags,,length of block |
| | | B0 AB%RED Break on read reference |
| | | B1 AB%WRT Break on write reference |
| | | B2 AB%XCT Break on an execute (instruction fetch) reference |
| | | B3 AB%SEC Break on this address in any section |
| 1 | .ABLOB | Lower bound address |
| 2 | .ABUPB | Lower bound address |

Read address break range - .ABRRG

This function is the same as .ABRED except it returns the current address break information for a range of addresses. Currently the range is restricted to a single address location. This function requires that AC2 contain the address of an argument block. The user

TOPS-20 MONITOR CALLS
(ADBRK)

fills in word 0; the monitor supplies the remaining information. The format of the argument block is:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .ABHDR | Length of the block |
| 1 | .ABLOB | Lower bound address (return) |
| 2 | .ABUPB | Upper bound address (return) |
| 3 | .ABFLG | Flags (return), same as those for .ABSRG |

Return address break data - .ABGBR

This function is the same as .ABGAD except the address on which the break occurred is an address within the break range provided by the user. AC2 contains the address of an argument block with the following format:

| Word | Symbol | Contents |
|------|--------|------------------------|
| 0 | .ABHDR | Length of the block |
| 1 | .ABBPC | Break PC (return) |
| 2 | .ABBAD | Break address (return) |

Generates an illegal instruction interrupt on error conditions below.

ADBRK ERROR MNEMONICS:

ABRKX1: Address break not available on this system
ARGX02: Invalid function
FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Activates specific software interrupt channels. (See Section 2.6.)

TOPS-20 MONITOR CALLS
(AIC)

ACCEPTS IN AC1: Process handle

AC2: 36-bit word
Bit n on means activate channel n

RETURNS +1: Always

The DIC monitor call can be used to deactivate specified software interrupt channels.

Generates an illegal instruction interrupt on error conditions below.

AIC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Allocates a device to a job or to the device pool of the monitor's resource allocator. A device under control of the monitor's resource allocator cannot be opened or assigned by any job other than the one to which it is currently allocated. When the allocated device is deassigned, it is returned to the monitor's resource allocator.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code (.ALCAL)

AC2: Device designator

AC3: Job number, -1, or -2

RETURNS +1: Failure, error code in AC1

+2: Success

If AC3 contains a job number, then the designated device is allocated to that job.

If AC3 contains -1, then the device is returned to the pool of devices available to all users of the system (the device is no longer allocated). This is the initial state of all devices.

If AC3 contains -2, then the device is assigned to the monitor resource allocator's pool of devices.

TOPS-20 MONITOR CALLS
(ALLOC)

Once a job assigns or opens a nonallocated device (a device not under control of the resource allocator), the resource allocator cannot take the device from the job. The resource allocator can allocate the device, however, to the job that currently has it. Then, when the job releases the device, the resource allocator gets control of the device.

When a job returns control of a device to the system resource allocator, the allocator receives an IPCF packet. The flag word (.IPCFL) of the packet descriptor block contains a code that indicates the message was sent by the monitor. This code is 1(.IPCCC) in the IP%CFC field (bits 30-32).

The first word of the IPCF packet data block contains .IPCSA, which means that the second and subsequent words contain designators for devices returned to the control of the resource allocator.

```
.IPCFL/<.IPCCC>B32  
  
DATA/.IPCSA  
DATA+1/device designator  
DATA+2/device designator
```

The ALLOC monitor call requires the process to have WHEEL or OPERATOR capability enabled.

ALLOC ERROR MNEMONICS:

```
ALCX1:   Invalid function  
ALCX2:   WHEEL or OPERATOR capability required  
ALCX3:   Device is not assignable  
ALCX4:   Invalid job number  
ALCX5:   Device already assigned to another job  
ALCX6:   Device assigned to user job, but will be given to allocator  
         when released  
DEVX1:   Invalid device designator
```

Performs operations pertaining to the archive and virtual disk systems. These include requesting archival and migration, requesting retrieval, and setting archive status and tape information for a file.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: JFN

AC2: Function code.

TOPS-20 MONITOR CALLS
(ARCF)

The available functions and their argument blocks are described below.

AC3: (Function-dependent, normally 0)

| Code | Symbol | Function |
|------|-----------|--|
| 0 | .ARRAR | Sets/clears AR%RAR (in .FBBBT of the FDB), activating or deactivating a user request for archival. The value .ARSET (1) in AC3 requests an archive while .ARCLR (0) clears the request. Specifying .ARSET in AC3 sets AR%NDL (in .FBBBT of the FDB) and requests that the contents of the file not be flushed from disk upon archival. |
| 1 | .ARRIV | Sets/clears AR%RIV (in .FBBBT of the FDB), activating or deactivating a system request to migrate a file from disk to tape. The value .ARSET in AC3 requests migration while .ARCLR clears the request. This function requires WHEEL or OPERATOR capability to be enabled. |
| 2 | .AREXM | Sets/clears AR%EXM (in .FBBBT of the FDB), activating or deactivating exemption from involuntary migration. Code .ARSET (1) in AC3 sets AR%EXM, while code .ARCLR (0) in AC3 clears AR%EXM. This function requires WHEEL or OPERATOR capability to be enabled. |
| 3 | .ARRFR | Request that the contents of a file be restored to disk. The contents of AC3 determine if .ARRFR waits or returns without waiting for the contents of the file to be restored to disk. |
| | | Options for AC3 |
| | B0 AR%NMS | Do not wait for the file to be restored. |
| | B1 AR%WAT | Wait until the file is restored. |
| 4 | .ARDIS | Discard tape information. Clears FB%ARC (if set), .FBTP1, .FBTP2, .FBTSN, .FBTFN, and .FBTDT. The file must be on line for the function to succeed. Options for AC3 (which require WHEEL or OPERATOR capability enabled to be used separately): |
| | B0 AR%CR1 | Clear information for run 1. |
| | B1 AR%CR2 | Clear information for run 2. |

TOPS-20 MONITOR CALLS
(ARCF)

5 .ARSST Set tape information for a file. This function is used to set information for the first, second, or both tape runs. AR%01 and AR%02 are used together when restoring files to disk. It requires enabled WHEEL or OPERATOR privileges.

AC3 contains a pointer to an argument block as follows:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .AROFL | Flags: |
| | | B0(AR%01) Set information for run 1. |
| | | B1(AR%02) Set information for run 2. |
| | | B2(AR%OFL) Delete disk contents of file when done. Requires both run 1 and run 2 tape information to be set. |
| | | B3(AR%ARC) Set FB%ARC in the FDB (archive the file.) |
| | | B4(AR%CRQ) Clear archive and/or migration requests (clear AR%RAR and AR%RIV.) |
| 1 | .ARTP1 | Tape 1 identification. |
| 2 | .ARSF1 | TSN 1,,TFN 1 - Tape saveset number in the left half and tape file number in the right half. |
| 3 | .ARTP2 | Tape 2 identification. |
| 4 | .ARSF2 | TSN 2,,TFN 2 - similar to .ARSF1. |
| 5 | .ARODT | time and date of tape write in internal format; 0 implies present time. |

TOPS-20 MONITOR CALLS
(ARCF)

| | | |
|----|--------|---|
| 6 | .ARPSZ | Number of pages in the file. This word can be set only if AR%01 and AR%02 are set first. |
| 6 | .ARRST | Restore contents of a file to disk. AC3 contains a JFN for a temporary file (created by DUMPER) that contains the data for an archived file that is currently off-line. After .FBADR, .FBBSY, and .FBSIZ are copied, the temporary file is deleted. Both files must be on the same device or structure, and enabled WHEEL or OPERATOR capability is required. |
| 7 | .ARGST | Get tape information for file. AC3 contains the address of an argument block that has the same format as the block for .ARSST. |
| 10 | .ARRFL | The restore for this file has failed. Sets AR%RFL in .FBBBT to notify a waiting process that the retrieval request cannot be completed. Requires WHEEL or OPERATOR capability. |
| 11 | .ARNAR | Resist involuntary migration. Sets or clears AR%NAR in .FBBBT. Using .ARSET in AC3 causes resist to be set, while using .ARCLR clears resist. |

ARCF ERROR MNEMONICS:

| | |
|---------|--|
| CAPX1: | WHEEL or OPERATOR capability required |
| ARGX02: | Invalid function code |
| ARCFX2: | File already has archive status |
| ARCFX3: | Cannot perform ARCF functions on nonmultiple directory devices |
| ARCFX4: | File is not on line |
| ARCFX5: | Files are not on the same device or structure |
| ARCFX6: | File does not have archive status |
| ARCFX7: | Invalid parameter for .ARSST |
| ARCFX8: | Archive not complete |
| ARCFX9: | File not off line |
| ARCX10: | Archive prohibited |
| ARCH11: | Archive requested, modification prohibited |
| ARCH12: | Archive requested, delete prohibited |
| ARCX13: | Archive system request not completed |
| ARCX14: | Restore failed |
| ARCX15: | Migration prohibited |
| ARCX16: | Cannot exempt off-line file |
| ARCX17: | FDB improper format for ARCF |
| ARCX18: | Retrieval wait cannot be fulfilled for waiting process |
| ARCX19: | Migration already pending |

TOPS-20 MONITOR CALLS
(ASND)

Assigns a device to the caller. The successful return is given if the device is already assigned to the caller.

ACCEPTS IN AC1: Device designator

RETURNS +1: Failure, error code in AC1

+2: Success

The RELD call can be used to release devices assigned to the caller.

ASND ERROR MNEMONICS:

DEVX1: Invalid device designator
DEVX2: Device already assigned to another job
ASNDX1: Device is not assignable
ASNDX2: Illegal to assign this device
ASNDX3: No such device
DSMX1: File(s) not closed

Assigns Internet queues for the TCP/IP interface.

RESTRICTIONS: For TCP/IP systems only. Requires NET WIZARD, WHEEL, or OPERATOR capability enabled.

ACCEPTS IN AC1: Flags in the left half and a pointer to the Queue Descriptor Block in the right half.

AC2: Unused, must be 0

AC3: Unused, must be 0

RETURNS +1: Failure, with error code in AC1 and conflicting job number in AC2

+2: Success, with internet queue handle in AC1 and the maximum SNDIN% count in AC2

ASNIQ% Flags

| Bit | Symbol | Meaning |
|-----|--------|----------------------|
| B1 | AQ%SPT | Single-port protocol |

TOPS-20 MONITOR CALLS
(ASNIQ%)

B2 AQ%ICM Deliver ICMP error datagrams to this queue

Queue Descriptor Block Format:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .IQPRV | B0-23 Must be 0 B24-31 Internet protocol number |
| 1 | .IQFHV | B0-31 Internet foreign host value word |
| 2 | .IQSHV | B0-31 Internet source host value word; used for logical host selection |
| 3 | .IQPTV | Internet port value word B0-15 Local port value B16-31 Foreign port value; ignored if bit AQ%SPT is set |
| 4 | .IQPRM | Mask word corresponding to .IQPRV |
| 5 | .IQFHM | Mask word corresponding to .IQFHV |
| 6 | .IQSHM | Mask word corresponding to .IQSHV |
| 7 | .IQPTM | Mask word corresponding to .IQPTV; use 0 for portless protocols |
| 8 | .IQLEN | Length of argument block. |

Mask words specify those bit positions where an exact match is required. Note that an error will occur unless the current Queue Descriptor Block differs in masked bits from all other Internet queues which are assigned at the time the ASNIQ% JSYS is executed.

ASNIQ% ERROR MNEMONICS:

ARGX22: Invalid flags

Assigns a special message queue to a job.

RESTRICTIONS: For TCP/IP systems only. Requires NET WIZARD
 capability (SC%NWZ).

ACCEPTS IN AC1: Mask

TOPS-20 MONITOR CALLS
(ASNSQ)

AC2: Header value

RETURNS +1: Failure, error code in AC1

 +2: Success, special message queue assigned with special
 queue handle in AC1

ASNSQ ERROR MNEMONICS:

NTWZX1: NET WIZARD capability required
ASNSX1: Insufficient system resources (All special queues in use)
ASNSX2: Link(s) assigned to another special queue

Detaches the specified job from its controlling terminal (if any) and optionally attaches it to a new controlling terminal. A console-attached entry is appended to the accounting data file.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: B0(AT%CCJ) Generate a CTRL/C interrupt to the lowest process in the job that is enabled for a CTRL/C interrupt if the job is currently attached to another terminal. If this bit is not set or if the job is currently not attached to another terminal, the job simply continues running when it is attached.

B1(AT%NAT) Do not attach. Prevents both the detaching of the job from its terminal and the attaching of a remote job to the local terminal. Is a no-op unless the remote job has a controlling terminal, in which case the remote job is detached and remains detached. This bit in effect makes ATACH like a remote DTACH.

B2(AT%TRM) Attach the given job to the terminal specified in AC4. If this bit is not set, the job is attached to the controlling terminal of the caller.

B18-35 Job number of the desired job.
(AT%JOB)

TOPS-20 MONITOR CALLS
(ATACH)

AC2: User number under which the job to be attached is logged in. The user number can be obtained with the RCUSR monitor call.

AC3: Byte pointer to an ASCIIZ password string in the caller's address space.

AC4: Number of the terminal to be attached to the specified job. This argument is required if B2(AT%TRM) is set.

RETURNS +1: Failure, error code in AC1.

+2: Success. If there is a logged-in job currently attached to the specified terminal, it is detached and primary I/O for that job is not redirected. Thus, if a process has primary I/O from the controlling terminal, it will block when it attempts primary I/O and will continue when it is reattached and a character is typed. A job attached to the terminal but not logged in is killed.

It is legal to attach to a job that has a controlling terminal if one of the following conditions exists:

1. The job is logged in under the same user name as the job executing the ATACH.
2. The job executing the ATACH supplies the correct password of the job it is attaching to.
3. The job executing the ATACH has WHEEL or OPERATOR capability enabled.
4. The job executing the ATACH has ownership of the job because it created the job (and maintained ownership) with the CRJOB call.

If the controlling terminal is a PTY, a password is not required in the following cases:

1. The owner of the PTY has WHEEL or OPERATOR capability enabled.
2. The specified job is logged in with the same name as the owner of the PTY.

The DTACH monitor call can be used to detach the controlling terminal from the current job.

TOPS-20 MONITOR CALLS
(ATACH)

ATACH ERROR MNEMONICS:

ATACX1: Invalid job number
ATACX2: Job already attached
ATACX3: Incorrect user number
ATACX4: Invalid password
ATACX5: This job has no controlling terminal
ATACX6: Terminal is already attached to a job
ATACX7: Illegal terminal number

Assigns a terminal code to a software interrupt channel. (Refer to Section 2.6.) This call also sets the corresponding bit in the process's terminal interrupt mask. (Refer to the STIW and RTIW monitor calls.)

ACCEPTS IN AC1: Terminal interrupt code,,channel number
(Refer to Section 2.6.6.)

RETURNS +1: Always

If there is no controlling terminal (if the job is detached), the assignments are remembered and are in effect when a terminal becomes attached.

The DTI monitor call can be used to deassign a terminal code.

Generates an illegal instruction interrupt on error conditions below.

ATI ERROR MNEMONICS:

TERMX1: Invalid terminal code
ATIX1: Invalid software interrupt channel number
ATIX2: Control-C capability required

Creates the Network Virtual Terminal (NVT) connection.

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: Flag bits in the left half and the JFN of the opened receive connection in the right half

AC2: JFN of the opened send connection

TOPS-20 MONITOR CALLS
(ATNVT)

RETURNS +1: Failure, with error code in AC1
 +2: Success, with terminal designator specific to this
 NVT in AC1

Flags for AC1:

| Bit | Symbol | Meaning |
|-----|--------|--|
| B0 | AN%TCP | If set, this bit indicates that the right half of AC1 contains the TCP JCN instead of a JFN. |

ATNVT ERROR MNEMONICS:

ATNX1: Invalid receive JFN
ATNX2: Receive JFN is not open for read
ATNX3: Receive JFN is not open
ATNX4: Receive JFN is not a network connection
ATNX5: Receive JFN has been used
ATNX6: Receive connection has been refused
ATNX7: Invalid send JFN
ATNX8: Send JFN is not open for write
ATNX9: Send JFN is not open
ATNX10: Send JFN is not a network connection
ATNX11: Send JFN has been used
ATNX12: Send connection has been refused
ATNX13: Insufficient system resources (no NVTs)

Inputs the next byte from the specified source. When the byte is read from a file, the file must first be opened, and the size of the byte given, with the OPENF call. When the byte is read from memory, a pointer to the byte is given. This pointer is updated after the call.

ACCEPTS IN AC1: Source designator

RETURNS +1: Always, with the byte right-justified in AC2

If the end of the file is reached, AC2 contains 0 instead of a byte. The program can process this end-of-file condition if an ERJMP or ERCAL is the next instruction following the BIN call.

The BOUT monitor call can be used to output a byte sequentially to a destination.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

TOPS-20 MONITOR CALLS
(BIN)

BIN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX1: File is not open for reading
IOX4: End of file reached
IOX5: Device or data error

Backs up the source designator's pointer by one byte.

ACCEPTS IN AC1: Source designator

RETURNS +1: Failure, error code in AC1
+2: Success, updated string pointer in AC1, if pertinent.
(This return actually decrements the pointer.)

The BKJFN call, when referring to a terminal, can be executed only once per TTY to back up one character. The BKJFN call cannot be issued again for the same TTY unless the input buffer has been cleared (with the CFIBF JSYS) or an input JSYS is executed for the TTY.

BKJFN, when referring to other designators, can be executed more than once in succession.

This call cannot be used with the DECnet devices SRV: or DCN:.

BKJFN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
BKJFX1: Illegal to back up terminal pointer twice
SFPTX2: Illegal to reset pointer for this file
SFPTX3: Invalid byte number
TTYX01: Line is not active

Performs basic maintenance and utility functions required for loading and dumping communications software. The TOPS-20 system process that performs these functions uses a DIGITAL-supplied protocol to perform them.

TOPS-20 MONITOR CALLS
(BOOT)

On KL10 Model B hardware, the BOOT JSYS is used to load and dump a PDP-11 connected to a DTE20.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled. Some functions are hardware specific.

ACCEPTS IN AC1: Function code

AC2: Address of argument block

RETURNS +1: Always

The available functions and their argument blocks are described below.

| Code | Symbol | Meaning |
|------|----------|---|
| 0 | .BTROM | Activate the hardware ROM bootstrap in the communications front end. |
| | | Argument Block: |
| | 0 .BTDTE | DTE-20 number |
| | 1 .BTERR | Error status flags returned on failure of the call |
| 1 | .BTLDS | Load a secondary bootstrap program into the communications front end. The secondary bootstrap, with a maximum size of 256 PDP-11 words, is loaded using the ROM bootstrap. The data to be loaded must be packed as two 16-bit PDP-11 words left justified in each 36-bit word. The entire bootstrap program must be loaded at once, and the caller blocks until the transfer is complete. |
| | | Argument Block: |
| | 0 .BTDTE | DTE-20 number |
| | 1 .BTERR | Error status flags returned on failure of the call |
| | 2 .BTSEC | Address of bootstrap program to be loaded |
| 2 | .BTLOD | Load the communications front-end memory using the previously loaded secondary or tertiary bootstrap program. The bootstrap program in the front end must abide by the protocol for DTE-20 transfers: |

TOPS-20 MONITOR CALLS
(BOOT)

the first two bytes of data supplied by the caller must be a count of the remaining number of data bytes.

Argument Block:

| | | |
|---|--------|--|
| 0 | .BTDTE | DTE-20 number |
| 1 | .BTERR | Error status flags returned on failure of the call |
| 2 | | Not used and must be zero |
| 3 | .BTFLG | User-supplied flag word. This word is not used and must be zero. |
| 4 | .BTCNT | Number of bytes to transfer |
| 5 | .BDTPT | Pointer to where the data is to be dumped in TOPS-20 |

4 .BTIPR Initialize the protocol to be used with this communications front end. After successful execution of this function, TOPS-20 processes interrupts from the given DTE-20.

Argument Block:

| | | |
|---|--------|---|
| 0 | .BTDTE | DTE-20 number |
| 1 | .BTPRV | Version number of the protocol to be used |

Protocol types:

| Symbol | Meaning |
|------------|---------------------|
| .VN20F (0) | RSX20F protocol |
| .VNMCB (1) | MCB DECNET protocol |

5 .BTTPR Stop the protocol currently running on this communications front end or line. Stop the protocol currently running on this communications front end or line. After successful execution of this function, TOPS-20 ignores interrupts from the given DTE-20 or line.

Argument Block:

| | | |
|---|--------|---------------|
| 0 | .BTDTE | DTE-20 number |
|---|--------|---------------|

TOPS-20 MONITOR CALLS
(BOOT)

6 .BTSTS Return the status type of the protocol running on the communications front end to the specified DTE or line. Also returns the name of the adjacent DECNET node for this front end.

Argument Block:

0 .BDTE DTE-20 number

1 .BTCOD Returned protocol version type. If no protocol is running, this word contains -1.

Protocol types:

| Symbol | Meaning |
|------------|---------------------|
| .VN20F (0) | RSX20F protocol |
| .VNMCB (1) | MCB DECNET protocol |

7 .BTBEL Block until a signal (doorbell) to TOPS-20 is initiated by the communications front end. This function is used to synchronize the caller with the bootstrap program in the front end.

Argument Block:

0 .BDTE DTE-20 number

10 .BTRMP Read data from the communications front end using the previously loaded secondary or tertiary bootstrap program. The bootstrap program must abide by the protocol for DTE-20 transfers. The first two bytes of data are interpreted as a count of the remaining number of bytes of data.

Argument Block:

0 .BDTE DTE-20 number

1 .BTERR Error status flags returned on failure of the call

2 Not used and must be zero

3 .BTFLG User-supplied flag word

B0(BT%BEL) Send a signal (doorbell) to TOPS-20 to indicate the transfer is finished.

TOPS-20 MONITOR CALLS
(BOOT)

| | | |
|----|--------|---|
| 4 | .BTCNT | Maximum number of bytes to transfer. After successful execution of this function, this word is updated to reflect the actual number of bytes transferred. |
| 5 | .BTMPT | Pointer to where data is to be placed |
| 14 | .BTCLI | Convert line id to port number Argument Block: 0 .BTPRT Port number 1 .BTLID Pointer to ASCIZ line id |
| 15 | .BTCPN | Convert NSP port number to line id Argument Block: 0 .BTPRT Port number 1 .BTLID Pointer to ASCIZ line id |
| 16 | .BTD60 | Send a message to or receive a message from a front end (a DN60) using the .VND60 protocol. The argument block controls whether this function sends or receives a message. (Requires DN60) Argument Block: 0 .BT6DTE DTE number 1 .BT6ERR Error flags (returned): 30 D6%BDP The data byte pointer passed in the argument block is bad. 31 D6%ARD The PDP-11 attempted to send data when none was expected. 32 D6%TRS DTESRV timed out waiting for response header from the front end. 33 D6%TDT DTESRV timed out waiting for data from the front end. 34 D6%TPO DTESRV timed out waiting for the DTE to be free. Another job is using the DTE and is probably hung. |

TOPS-20 MONITOR CALLS
(BOOT)

| | | | | |
|----|---------|----|--------|--|
| | | 35 | D6%NT6 | The front end is not running DN60 protocol. |
| 2 | .BT6HBC | | | Number of bytes in the DN60 header. |
| 2 | .BT6HDR | | | Address at which the DN60 header begins. This header contains 4 words, which contain 4 8-bit bytes each. |
| 3 | .BT6DBC | | | Number of bytes of data. |
| 4 | .BT6PTR | | | Pointer to the first byte of the data. |
| 5 | .BT6TMR | | | Time the request was made (returned). |
| 6 | .BT6TAS | | | Time DTE was assigned (returned). |
| 7 | .BT6THQ | | | Time TOPS-20 queued the header to the DTE. |
| 10 | .BT6TRD | | | Time TOPS-20 was done for response header. |
| 11 | .BT6TDD | | | Time TOPS-20 was done for data. |
| 12 | .BT6TFR | | | Time TOPS-20 satisfied the request. |

The error status flag returned in word .BTERR on failure of a BOOT call contains front-end reload status bits recorded in the system error file. Refer to the SPEAR manual for an explanation of these status bits. Note that error logging is not performed for group A processors.

Generates an illegal instruction interrupt on error conditions below.

BOOT ERROR MNEMONICS:

BOTX01: For group A processors, this message indicates an illegal line number. For group B processors, this message indicates an invalid DTE-20 number.

BOTX02: Invalid byte size

BOTX03: Invalid protocol version number

BOTX04: Byte count is not positive

BOTX05: Protocol initialization failed

BOTX06: GTJFN failed for dump file

BOTX07: OPENF failed for dump file

BOTX08: Dump failed

BOTX09: To -10 error on dump

BOTX10: To -11 error on dump

BOTX11: Failed to assign page on dump

BOTX12: Reload failed

BOTX13: -11 didn't power down

BOTX14: -11 didn't power up

BOTX15: ROM did not ACK the -10

BOTX16: -11 boot program did not make it to -11

TOPS-20 MONITOR CALLS
(BOOT)

BOTX17: -11 took more than 1 minute to reload; will cause retry
BOTX18: Unknown BOOT error
CAPX1: WHEEL or OPERATOR capability required
ARGX02: invalid function

Outputs a byte sequentially to the specified destination. When the byte is written to a file, the file must first be opened, and the size of the byte given, with the OPENF call. When the byte is written to memory, AC1 contains a pointer to the location in which to write the byte. This pointer is updated after the call.

ACCEPTS IN AC1: Destination designator
AC2: Byte to be output, right-justified

RETURNS +1: Always

The BIN monitor call can be used to input a byte sequentially from a source.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

BOUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX2: File is not open for writing
IOX5: Device or data error
IOX6: Illegal to write beyond absolute end-of-file
IOX11: Quota exceeded
IOX33: TTY input buffer full
IOX34: Disk full
IOX35: unable to allocate disk - structure damaged

Changes the account for the current job.

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

TOPS-20 MONITOR CALLS
(CACCT)

ACCEPTS IN AC1: Byte pointer that points to the new account string in the calling program's address space. This call reads the string until a null byte is read, or until 39 characters are read.

If executed in section 0, this AC can contain a local byte pointer or an account number. The account number must be in bits 3-35, and bits 0-2 must contain 5.

RETURNS +1: Failure, error code in AC1
+2: Success, updated string pointer in AC1

The CACCT call sets the current account for the job to the specified account. Subsequent session charges will be to this new account. This call also validates the account given if the account validation facility is enabled. (Refer to the .SFAVR function of the SMON/TMON monitor call.)

The GACCT monitor call can be used to return the account for the current job.

CACCT ERROR MNEMONICS:

CACTX1: Invalid account identifier
CACTX2: Job is not logged in
VACCX0: Invalid account
VACCX1: Account string exceeds 39 characters

Clears the designated file input buffer.

ACCEPTS IN AC1: Source designator

RETURNS +1: Always

Is a no-op if the source designator is not associated with a terminal.

The CFOBF monitor call can be used to clear a designated file output buffer.

Generates an illegal instruction interrupt on error conditions below.

CFIBF ERROR MNEMONICS:

DESX1: Invalid source/destination designator

TOPS-20 MONITOR CALLS
(CFIBF)

DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Clears the designated file output buffer.

ACCEPTS IN AC1: Destination designator

RETURNS +1: Always

Is a no-op if the destination designator is not associated with a terminal.

The CFIBF call can be used to clear a designated file input buffer.

Generates an illegal instruction interrupt on error conditions below.

CFOBF ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Creates a process inferior to the calling process. (Refer to Section 2.7.)

ACCEPTS IN AC1: Characteristics for inferior,,PC address for inferior

B0(CR%MAP) Make the inferior process's map the same as the current process's map by means of indirect pointers. If this bit is not on, the inferior process will have no pages in its map. If desired, the creating process can then use PMAP or GET to add pages to the inferior's map.

B1(CR%CAP) Make the inferior process's capabilities the same as the current process's. If this bit is not on, the inferior process

TOPS-20 MONITOR CALLS
(CFORK)

has no capabilities (all bits of Job Capability Word are 0).

B3(CR%ACS) Set the inferior process's ACs from the block whose address is in AC2. If this bit is not on, the inferior process's ACs are set to 0.

B4(CR%ST) Set the PC of the inferior process to the value in the right half of AC1 and start the process. If this bit is not on, the inferior process is not started, and the right half of AC1 is ignored. (Also see the XSFRK% call.)

B18-35 PC value for the inferior process if CR%ST (CR%PCV) is on.

AC2: Address of 20 (octal) word block (optional). This block contains the AC values for the inferior process. (Refer to bit CR%ACS above.)

RETURNS +1: Failure, error code in AC1
 +2: Success, relative process handle in AC1

The inferior process receives the same primary input and output JFNs as the current process. However, the primary input and/or output files may be changed with the SPJFN monitor call.

The CR%MAP argument in AC1 allows the inferior to see the same address space as that of the superior. The inferior process will have read and write access to the superior's address space. The pages are shared, and changes made by one process will be seen by the other.

CFORK creates a nonvirgin process if:

1. CR%ST is set and
2. CR%ACS and/or CR%MAP is set.

CFORK creates an execute-only process if bit CR%MAP is set and the creating process is an execute-only process. This is the only other way to create an execute-only process besides using the GET JSYS on a virgin process.

The KFORK monitor call can be used to kill one or more processes.

CFORK ERROR MNEMONICS:

FRKH6: All relative process handles in use

TOPS-20 MONITOR CALLS
(CFORK)

FRKHX8: Illegal to manipulate an execute-only process
CFRKX3: Insufficient system resources

Changes certain words in the file descriptor block (FDB) for the specified file. (Refer to Section 2.2.8 for the format of this block.)

RESTRICTIONS: WHEEL or OPERATOR capability required to change some words in the FDB. (Refer to Table 2-1 for the words requiring capabilities.)

ACCEPTS IN AC1: B0(CF%NUD) Do not wait for the disk copy of the directory to be updated.

The specified changes are made to the directory in memory and are written to the disk as a part of the normal monitor disk updating procedure. (See below for more information.)

B9-17 Index into FDB indicating word to be
(CF%DSP) changed

B18-35 JFN (for a disk file)
(CF%JFN)

AC2: Mask indicating bits to be changed. If changing a count value (in AC3), use -1 as a mask.

AC3: New values for changed bits. These values must be given in the bit positions corresponding to the mask given in AC2.

RETURNS +1: Always

Because each CHFDB call changes only one word in the FDB, several calls must be executed to change several words. Each call causes disk I/O. To keep I/O to a minimum, the program should set bit CF%NUD on each call. The setting of this bit on each call permits the program to run faster by allowing several changes to be made to the FDB with minimum disk I/O.

To ensure that all the changes have been written to the disk, the program can issue the last CHFDB call with bit CF%NUD off. Also, if the program requires the FDB on the disk to be updated after each call, it should execute each CHFDB call with bit CF%NUD off.

TOPS-20 MONITOR CALLS
(CHFDB)

There are a variety of calls used in manipulating the FDB; see the description of the FDB in Chapter 2 for information on these calls.

Generates an illegal instruction interrupt on error conditions below.

CHFDB ERROR MNEMONICS:

CFDBX1: Invalid displacement
CFDBX2: Illegal to change specified bits
CFDBX3: Write or owner access required
CFDBX4: Invalid value for specified bits
CFDBX5: No FDB for non-directory devices
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
STRX10: Structure is offline

Checks if a user is allowed access to files in a given directory. This monitor call determines if the user can access files that have a specified protection code if the user is logged in with the given capabilities and connected to the directory.

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Length of the argument block in the right half. If B0(CK%JFN) is on, word .CKAUD of the argument block contains a JFN.

AC2: Address of argument block

RETURNS +1: Failure, error code in AC1
+2: Success, access check is completed, with AC1 containing -1 if access is allowed or 0 if access is not allowed.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .CKAAC | Code of desired access to files. |
| 1 | .CKALD | Byte pointer to user name string, or 36-bit user number of user whose access is being checked. |

TOPS-20 MONITOR CALLS
(CHKAC)

- | | | |
|---|--------|--|
| 2 | .CKACD | Byte pointer to directory name string (with punctuation), or 36-bit directory number to which user whose access is being checked is connected. |
| 3 | .CKAEC | Enabled capabilities of user whose access is being checked. (Refer to Section 2.7.1.) |
| 4 | .CKAUD | Byte pointer to directory name string (with punctuation), or 36-bit directory number of the directory containing the files being accessed. If B0(CK%JFN) of AC1 is on, this word contains a JFN for the file being accessed. |
| 5 | .CKAPR | Protection of the files being accessed. (Refer to Section 2.2.6.) This word is not required if a JFN is supplied in word .CKAUD. |

Access codes are as follows:

- | | | |
|----|--------|--|
| 0 | .CKARD | read existing files |
| 1 | .CKAWR | write existing files |
| 2 | .CKAEX | execute existing files |
| 3 | .CKAAP | append to existing files |
| 4 | .CKADL | obtain directory listing of existing files |
| 6 | .CKADR | read the directory |
| 10 | .CKACN | connect to the directory |
| 11 | .CKACF | create files in the directory |

CHKAC ERROR MNEMONICS:

- | | |
|---------|--------------------------|
| CKAX1: | Argument block too small |
| CKAX2: | Invalid directory number |
| CKAX3: | Invalid access code |
| CKAX4: | File is not on disk |
| STRX10: | Structure is offline |

Clears the software interrupt system for the current process. Clears all interrupts in progress and all waiting interrupts.

RETURNS +1: Always

Closes a specific file or all files.

TOPS-20 MONITOR CALLS
(CLOSF)

ACCEPTS IN AC1: B0(CO%NRJ) Do not release the JFN.

B6(CZ%ABT) Abort any output operations currently being done. Close the file but do not perform any cleanup operations normally associated with closing a file. (If output is to a magnetic tape, for example, do not output remaining buffers or write tape marks. If output is to a disk file, do not change the end-of-file pointer.) If output is to a new disk file that has not been closed (and is therefore nonexistent), the file is closed and then expunged.

B7(CZ%NUD) Do not update the copy of the directory on the disk. (Refer to CF%NUD of the CHFDB call description for further information.)

B18-35 JFN of the file being closed
(CO%JFN)

RETURNS +1: Failure, error code in AC1

+2: Success

If AC1 contains -1, all files (and all JFNs) at or below this process (with the exception of the primary I/O files and files that cannot be closed by this process) are closed. This action is identical to that taken on a CLZFF call with AC1 containing the process handle .FHSLF (400000).

The OPENF monitor call can be used to open a specific file.

CLOSF ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
CLSX1: File is not open
CLSX2: File cannot be closed by this process
CLSX3: File still mapped
CLSX4: Device still active
ENQX20: Locked JFN cannot be closed
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

All output errors can occur.

TOPS-20 MONITOR CALLS
(CLZFF)

Closes process's files. Closes all files and/or releases all JFNs at and/or below a specified process.

ACCEPTS IN AC1: B0(CZ%NIF) Do not close files of inferior. processes

B1(CZ%NSF) Do not close files of this process.

B2(CZ%NRJ) Do not release JFNs.

B3(CZ%NCL) Do not close any files; only release nonopen JFNs

B4(CZ%UNR) Unrestrict files opened with restricted access for specified process. The specified process must be the same as, or inferior to, the process executing the call.

B5(CZ%ARJ) Wait until file can be closed, then close it, and release JFNs.

B6(CZ%ABT) Abort any output operations currently being done. Close the file but do not perform any cleanup operations normally associated with closing a file (for example, do not output remaining buffers or write tape marks if output to a magnetic tape is aborted). If output to a new disk file that has not been closed (file is nonexistent) is aborted, the file is closed and then expunged.

B7(CZ%NUD) Do not update the copy of the directory on the disk. (Refer to CF%NUD of the CHFDB call description for further information.)

B18-35 Process handle
(CZ%PRH)

RETURNS +1: Always. No action is taken if the call is in any way illegal.

If AC1 contains only the process handle .FHSLF, the action is identical to that taken on a CLOSF call with AC1 containing -1.

Generates an illegal instruction interrupt on error conditions below.

CLZFF ERROR MNEMONICS:

TOPS-20 MONITOR CALLS
(CLZFF)

FRKH1: Invalid process handle
 FRKH2: Illegal to manipulate a superior process
 FRKH3: Invalid use of multiple process handle
 IOX11: Quota exceeded
 IOX34: Disk full
 IOX35: Unable to allocate disk - structure damaged

Returns configuration information about the central processor and operating system environment for the system on which the monitor call is executed.

ACCEPTS IN AC1: Function code
 AC2: Address of argument block

RETURNS +1: Always

The available functions and their argument blocks are described below.

| Code | Symbol | Meaning |
|------|----------|---|
| 0 | .CFINF | Return basic hardware and software information. |
| | | Argument Block: |
| | 0 .CFLEN | Number of words returned (CF%WDP),, length of argument block (CF%LOB) |
| | 1 .CFIPR | Type of processor. ID for KL = .CFGKL(4) |
| | 2 .CFISE | CPU serial number, right-justified |
| | 3 .CFIUC | CPU microcode version number, right justified |
| | 4 .CFIHO | CPU hardware options: |
| | | B0(CF%50Z) Line power is 50 hertz. |
| | | B1(CF%CHI) Cache is installed. |
| | | B2(CF%CHN) Channel bit in the APRID word is on. |
| | | B3(CF%EKL) CPU is an extended KL10. |

TOPS-20 MONITOR CALLS
(CNFIG)

B4(CF%MOS) System has a master oscillator.

B5(CF%MCA) System has MCA25 pager cache.

B6(CF%CH1) Cache control bit 1.

B7(CF%CH2) Cache control bit 2.

B8(CF%CI) System has a CI.

5 .CFIMO CPU microcode options

B0(CF%T20) TOPS-20 paging implemented.

B1(CF%EAD) Microcode handles extended addresses.

B2(CF%UCO) Non-standard microcode is loaded.

6 .CFISO TOPS-20 static software options

B0(CF%CFS) CFS is installed.

B1(CF%DCN) DECnet is installed.

B2(CF%ARP) TCP/IP is installed.

7 .CFIVR TOPS-20 version number obtained from location .JBVER.

The maximum length of the argument block is given by symbol .CFLIN.

1 .CFCIN Return CFS information

Argument Block:

0 .CFLEN Number of words returned (CF%WDP),, length of argument block (CF%LOB).

1 .CFNCN Number of CFS nodes up, including the host system.

2 .CFCDO CFS dynamic options

B0(CF%CFR) Host has connected to another CFS host at least once.

TOPS-20 MONITOR CALLS
(CNFIG)

The maximum length of the argument block is given by symbol .CFCLN.

- 2 .CFCSE Return CI node number and serial number of each CFS node. The numbers are returned right justified in APRID format. Bits 0-13 of each word are reserved for the future by DIGITAL. Information will be returned for a host, provided that the host is active and that there is valid information for the host. Information for the first host will always be returned. The number of hosts is determined by word .CFNCN of the .CFCIN function.

Argument Block:

- | | | |
|---|--------|--|
| 0 | .CFLEN | Number of words returned (CF%WDP),, length of argument block (CF%LOB). |
| 1 | .CFCS1 | CI node number (CF%CIN),, serial number of first host (CF%HSN). |
| 2 | | CI node number (CF%CIN),, serial number of next host (CF%HSN). |
| n | .CFCSn | CI node number (CF%CIN),, serial number of last host (CF%HSN). |

- 3 .CFCND Return node names of CFS hosts as 2-word ASCIZ strings. Information will be returned for a host provided that the host is active and that there is valid information for the host. Information for the first host will always be returned. The number of hosts is determined by word .CFNCN of the .CFCIN function.

Argument Block:

- | | | |
|---|----------|--|
| 0 | .CFNND | Number of nodes returned (CF%NND),, length of argument block (CF%LOB). |
| 1 | .CFBP1 | Byte pointer to ASCIZ node name of first host. |
| | .CFBP1+n | Start of area where node name strings are placed. |

- 4 .CFHSC Returns the list of HSC node names. In the event that the argument block is not large enough, the

TOPS-20 MONITOR CALLS
(CNFIG)

CFGBTS error code is returned. Since the argument block must be long enough to contain all possible HSCs, it is suggested that it be set to the length C%SBL*3+1.

Argument Block:

| | | |
|---|----------|---|
| 0 | .CFNHN | Number of nodes returned (CF%NHN),,length of block (CF%LOB). |
| 1 | .CFHP1 | Byte pointer to first node name string |
| | .CFHP1+n | Start of an area in which the monitor placed node name strings. These are ASCIZ strings containing the node name. |

Generates an illegal instruction interrupt on error conditions below.

CNFIG% ERROR MNEMONICS:

CFGBFC: Function code out of range
CFGBTS: Argument block too short
CFGIAB: Invalid argument block address
CFGAAB: Error accessing argument block
CFGINA: Information not available for this function

Parses one field of a command that is either typed by a user or contained in a file. When this monitor call is used to read a command from a terminal, it provides the following features:

1. Allows the input of a command (including the guide words) to be given in abbreviated, recognition (ESC and CTRL/F), and/or full input mode.
2. Allows the user to edit his input with the DELETE, CTRL/U, CTRL/W, and CTRL/R editing keys.
3. Allows fields of the command to be defaulted if an ESC or CTRL/F is typed at the beginning of any field, or if a field is omitted entirely.
4. Allows a help message to be given if a question mark (?) is typed at the beginning of any field.

TOPS-20 MONITOR CALLS
(COMND)

5. Allows input of an indirect file (@file) that contains the fields for all or the remainder of the command.
6. Allows a recall of the correct portion of the last command (up to the beginning of the field where an error was detected) if the next command line begins with CTRL/H. The correct portion of the command is retyped, and the user can then continue typing from that point.
7. Allows input of a line to be continued onto the next line if the user types a hyphen (-) immediately preceding a carriage return. (The carriage return is invisible to the program executing the COMND call, although it is stored in the text buffer.) The user can type the hyphen while he is typing a comment. The comment is then continued onto the next line.

A hyphen not immediately followed by a carriage return is parsed as ordinary text.

The COMND call allows comments in the command line. A command line can contain a comment if the field before the comment has been terminated and the comment is preceded by an exclamation point or a semicolon. If the comment starts with an exclamation point, COMND ignores all text between the exclamation point and either the end of the line or the next exclamation point. If the comment starts with a semicolon, COMND ignores all text on the remainder of the line.

A command line can contain the name of an indirect command file so long as the file name comes at the beginning of a field. It must, however, be the last item on the line, and its contents must complete the command. The user must follow the name of the indirect command file (after any recognition is performed) with a carriage return.

If a carriage return does not end the command line immediately after the name of the indirect command file, the system outputs the message ?INDIRECT FILE NOT CONFIRMED. Also, if the user types a question mark (instead of the file specification of the indirect file) after he types the at-sign (@) character, the message FILESPEC OF INDIRECT FILE is output.

If the indirect file itself contains an ESC or a carriage return, COMND treats them as spaces. COMND places the contents of the indirect file in the text buffer, but does not display them on the user's terminal.

As the user types his command, the characters are placed in a command text buffer. This buffer can also include the command line prompt. Several byte pointers and counts reflect the current state of the parsing of the command. These pointers and counts are as follows:

TOPS-20 MONITOR CALLS
(COMND)

Normally, parsing does not begin, and the COMND call does not return control to the program, until an action character is typed. But if B8(CM%WKF) is on in word .CMFLG when the COMND call executes, parsing begins after each field is terminated.

A program parses a command line by repeated COMND calls. Each call specifies the type of field the program expects to be parsed. The program supplies this information, placing a function code and any data needed for the function in a function descriptor block. On successful completion of each call, the byte pointers and counts are updated in the command state block, and any data obtained for the field is returned.

The program executing the COMND call should not reset the byte pointers in the command state block after it completes parsing a command line. It should set up the command state block before it begins to parse any commands, and then use the .CMINI function to initialize the command state block before parsing each command line. This allows the .CMINI function to use the CTRL/H error-recovery feature.

If the program resets the pointers and counts in the command state block, instead of using the .CMINI function to do so, use of the CTRL/H feature is not possible. When a CTRL/H is typed, the .CMINI function allows recovery from an error in the last command only if the following are both true:

1. The pointer to the beginning of the user's input (.CMBFP) and the pointer to the beginning of the next field to be parsed (.CMPTR) are not equal.
2. The last character parsed in the previous command is not an end-of-line character.

The COMND call allows the user to delete his typed input with the DELETE, CTRL/W, and CTRL/U keys without regard to field boundaries. When the user deletes part of a field that has already been parsed, the COMND call returns to the program with B3(CM%RPT) set in word .CMFLG, or the program resumes execution at the reparse address contained in word .CMFLG of the command state block. This address should be the place in the program at which parsing of the command line begins. If this address is zero, the program must test AC1 for this bit, and reparse the command line from the beginning, if necessary. (See the description of word .CMFLG of the command state block.)

The calling sequence to the COMND call is as follows:

ACCEPTS IN AC1: Address of the command state block

AC2: Address of the first alternative function descriptor block

TOPS-20 MONITOR CALLS
(COMND)

RETURNS +1: Always (unless a reparse is needed and the right half of .CMFLG is nonzero), with

AC1 containing flags in the left half and the address of the command state block in the right half. The flags are copied from word .CMFLG in the command state block.

AC2 containing either the data obtained for the field or a monitor call error code if the field could not be parsed (CM%NOP is on in AC1).

AC3 containing in the left half the address of the function descriptor block given in the call, and in the right half the address of the function descriptor block actually used. Note that the contents of the right half identify uniquely the type of atom that was parsed.

The format of the command state block is shown below.

| | 0 | 17 18 | | 35 |
|--------|---|-------|---|------------------------------|
| | !=====! | | | |
| .CMFLG | ! Flag Bits ! | | ! | ! Reparse Dispatch Address ! |
| | !-----! | | | |
| .CMIOJ | ! Input JFN ! | | ! | ! Output JFN ! |
| | !-----! | | | |
| .CMRTY | ! Byte Pointer to CTRL/R Text ! | | | |
| | !-----! | | | |
| .CMBFP | ! Byte Pointer to Start of Text Buffer ! | | | |
| | !-----! | | | |
| .CMPTR | ! Byte Pointer to Next Input To Be Parsed ! | | | |
| | !-----! | | | |
| .CMCNT | ! Count of Space Left in Buffer ! | | | |
| | !-----! | | | |
| .CMINC | ! Count of Unparsed Characters in Buffer ! | | | |
| | !-----! | | | |
| .CMABP | ! Byte Pointer to Atom Buffer ! | | | |
| | !-----! | | | |
| .CMABC | ! Size of Atom Buffer ! | | | |
| | !-----! | | | |
| .CMGJB | ! Address of GTJFN Argument Block ! | | | |
| | !=====! | | | |

TOPS-20 MONITOR CALLS
(COMND)

Command State Block

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .CMFLG | <p>Flag bits in the left half, and the reparse dispatch address in the right half. Some flag bits can be set by the program executing the COMND call; others can be set by the COMND call after its execution. The bits that can be set by the program are described following the Command State Block description.</p> <p>The reparse dispatch address is the location to which control is transferred when a reparse of the command is needed. This happens when a user edits characters in a field that was already parsed.</p> <p>If this field is zero, the COMND call sets B3(CM%RPT) in the left half of this word, and gives the +1 return when a reparse is needed. The program must then test the left half of AC1 to see if CM%RPT is set. If it is, the user must reenter the code that parses the first field of the command.</p> <p>The code at the reparse dispatch address should initialize the program's state to what it was after the last .CMINI function. This initialization should include resetting the stack pointer, closing and releasing any JFNs acquired since the last .CMINI function, and transferring control to the code immediately following the last .CMINI function call.</p> |
| 1 | .CMIOJ | <p>Input JFN in the left half, and output JFN in the right half. These designators identify the source for the input of the command and the destination for the output of the typescript. These designators are usually .PRIIN (for input) and .PRIOU (for output).</p> |
| 2 | .CMRTY | <p>Byte pointer to the beginning of the prompting text.</p> |
| 3 | .CMBFP | <p>Byte pointer to the beginning of the user's input. The user cannot edit back past this pointer.</p> |
| 4 | .CMPTR | <p>Byte pointer to the beginning of the next field to be parsed.</p> |
| 5 | .CMCNT | <p>Count of the space remaining in the buffer after the .CMPTR pointer.</p> |

TOPS-20 MONITOR CALLS
(COMND)

- 6 .CMINC Count of the number of unparsed characters in the buffer after the .CMPTR pointer.

- 7 .CMABP Byte pointer to the atom buffer, a temporary storage buffer that contains the last field parsed by the COMND call. The terminator of the field is not placed in this buffer. The atom buffer is terminated with a null.

- 10 .CMABC The size of the atom buffer in bytes. The atom buffer should be at least as large as the largest field the program must parse.

- 11 .CMGJB Address of a GTJFN argument block. This block must be at least 16(octal) words long and must be writable. If a longer GTJFN block is being reserved, the count in the right half of word .GJF2 of the GTJFN argument block must be greater than four.

The GTJFN block is filled in by the COMND call with arguments for the GTJFN call if the specified COMND function requests a JFN (functions .CMIFI, .CMOFI, and .CMFIL). The user should store data in this block on the .CMFIL function only.

The flag bits that can be set by the user in the left half of word .CMFLG in the Command State Block are described below. These bits apply to the parsing of the entire command and are preserved by COMND after execution. See the end of the COMND JSYS discussion for the bits that are returned by COMND in the left half of word .CMFLG.

Bits Supplied in State Block on COMND Call

| Bit | Symbol | Meaning |
|-----|--------|---|
| 6 | CM%RAI | Convert lowercase input to uppercase. |
| 7 | CM%XIF | Do not recognize the at-sign (@) character as designating an indirect file; instead consider the character as ordinary punctuation. A program sets this bit to prevent the input of an indirect file. |
| 8 | CM%WKF | Begin parsing after each field is terminated instead of only after an action character (carriage return, ESC, CTRL/F, question mark) is typed. A program sets this bit if it must change terminal characteristics in the middle of a command. Turning off echoing during the input of a password is an example of a use for this bit. |

TOPS-20 MONITOR CALLS
(COMND)

Use of this bit is not recommended, however, because terminal wakeup occurs after each field is terminated, thereby increasing system overhead.

The recommended method of changing terminal characteristics within a command is to input the field requiring the special characteristic on the next line with its own prompt. For example, if a program is accepting a password, it should turn off echoing after the .CMCFM function of the main command and perform the .CMINI function to type the prompt requesting a password on the next line.

The format of the function descriptor block is shown below.

```

      0           8 9           17 18           35
      !=====!
      ! function ! function ! address of next function !
      ! code     ! flags   ! descriptor block   !
.CMFNP!-----!
.CMDAT!           Data for specific function           !
!-----!
.CMHLP!           Byte pointer to help text for field   !
!-----!
.CMDEF!           Byte pointer to default string for field !
!-----!
.CMBRK!           Address of 4-word break mask           !
!=====!

```

Function Descriptor Block

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .CMFNP | Function code and pointer to next function descriptor block. B0-8(CM%FNC) Function code B9-17(CM%FFL) Function-specific flags B18-35(CM%LST) Address of the next function descriptor block, or zero if this is the last function descriptor block. |
| 1 | .CMDAT | Data for the specific function, if any. |
| 2 | .CMHLP | Byte pointer to the help text for this field. This word can be zero if the program is not supplying its own help text. CM%HPP must be set (in word 0) in order for this pointer to be used. |

TOPS-20 MONITOR CALLS
(COMND)

- | | | |
|---|--------|--|
| 3 | .CMDEF | Byte pointer to the default string for this field. This word can be zero if the program is not supplying its own default string. CM%DPP must be on in word 0 in order for this pointer to be used. |
| 4 | .CMBRK | Address of a 4-word break mask that specifies which characters terminate a field. Word .CMBRK is ignored unless CM%BRK (B13) is on in word 0 of the function descriptor block. |

The individual words in the function descriptor block are described in the following paragraphs.

Words .CMFNP and .CMDAT of the function descriptor block

Word .CMFNP contains the function code for the field to be parsed, and word .CMDAT contains any additional data needed for that function. The function codes, along with any required data for the functions, are described below.

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .CMKEY | Parse a keyword, such as a command name. Word .CMDAT contains the address of a keyword symbol table. The keyword table must be in alphabetical order. See the TBLUK monitor call description for more information on the format of the keyword table. |

The table entries point to argument blocks. The right half of the first word of each such block contains the following bits, which can be set when B0-6 of that first word are off and B7(CM%FW) is set:

| | |
|-------------|---|
| B35(CM%INV) | Suppress this keyword in the list output on a question-mark (?). The program can set this bit to include entries in the table that should be output as part of the help text because they are not preferred keywords. This bit is also used with the CM%ABR bit to prevent an abbreviation from being output when a question mark (?) is typed. |
|-------------|---|

This bit can be set, for example, to allow the keyword LIST to be valid, even though the preferred keyword may be PRINT. The LIST keyword is not listed in the output given when a question mark (?) is typed.

TOPS-20 MONITOR CALLS
(COMND)

B34(CM%NOR) Do not recognize this keyword even if an exact match is typed by the user and suppress its listing in the list output when a question mark (?) is typed. (Refer to the TBLUK call description for more information on using this bit.)

B33(CM%ABR) Consider this keyword a valid abbreviation for another entry in the table. The right half of this table entry points to the command table entry of the keyword for which this is an abbreviation. The program can set this bit to include entries in the table that are less than the minimum unique abbreviation.

For example, this bit can be set to include the entry ST (for START) in the table. If the user then types ST as a keyword, COMND accepts it as a valid abbreviation for START even though there may be other keywords beginning with ST.

To suppress the output of this abbreviation in the list of keywords output when a question mark (?) is typed, the program must also set the CM%INV bit.

On a successful return, AC2 contains the address of the table entry where the keyword was found.

Note that keywords in the table that contain trailing spaces (such as FORTRAN literals) are not recognized.

- | | | |
|---|--------|--|
| 1 | .CMNUM | Parse a number. Word .CMDAT contains the radix (from 2 to 10) of the number. On a successful return, AC2 contains the number. |
| 2 | .CMNOI | Parse a guide word string, but do not return an error if no guide word is input. Guide words are output if the user terminated the previous field with ESC. Guide words are not output, nor can they be input, if the user has caused parsing into the next field. |

TOPS-20 MONITOR CALLS
(COMND)

For COMND to input a guide word, the guide word field must be delimited by parentheses. Word .CMDAT contains a byte pointer to an ASCIIZ string that contains the guide word. This string does not contain parentheses.

An error is returned only if a guide word is input that does not match the one expected by the COMND call.

- 3 .CMSWI Parse a switch. A switch field must begin with a slash, and can end with a colon or any legal field terminator.

Word .CMDAT contains the address of a switch keyword symbol table. (Refer to the TBLUK monitor call description for the format of the table.) Switch entries in the keyword table must not contain a slash. If switch requires a value, however, its entry must end with a colon.

The data bits CM%INV, CM%NOR, and CM%ABR, defined for the .CMKEY function, can also be set on this function.

On a successful return, AC2 contains the address of the table entry where the switch keyword was found.

- 4 .CMIFI Parse an input file specification. This function causes the COMND call to execute a GTJFN call, which attempts to parse the specification for an existing file using no default fields. Hyphens in the file specification are treated as alphanumeric characters.

The .CMGJB address (word 11 in the command state block) must be supplied, but the GTJFN block should be empty. Data stored in the GTJFN block is overwritten by the COMND JSYS, and GTJFN flags are set in the GTJFN block.

On a successful return, AC2 contains the JFN assigned.

See note following .CMFIL function.

- 5 .CMOFI Parse an output file specification. This function causes the COMND call to execute a GTJFN call, which parses the specification for either a new or an existing file. The default generation number

TOPS-20 MONITOR CALLS
(COMND)

is the generation number of the existing file plus 1. The .CMGJB address must be supplied, but the GTJFN block should be empty. (Data stored in the block will be overwritten by the COMND JSYS. Also, certain GTJFN flags are set.) On a successful return, AC2 contains the JFN assigned. Hyphens are treated as alphanumeric characters for this function.

See note following .CMFIL function.

6 .CMFIL Parse a general (arbitrary) file specification. This function causes the COMND call to execute a GTJFN to attempt to parse the specification for the file. The .CMGJB address must be supplied, but data stored in certain words of the GTJFN block is overwritten by the COMND JSYS and certain GTJFN flags are set (see note below). On a successful return, AC2 contains the JFN assigned. Hyphens are treated as alphanumeric characters for this function.

Note that portions of the GTJFN block used by functions .CMOFI, .CMIFI, and .CMFIL are controlled by COMND. The following list shows which words are under the control of COMND and which words are under the control of the user:

| GTJFN Word(s) | Controlled by | Characteristics |
|--------------------|------------------|---|
| .GJGEN | COMND | 1. .CMOFI sets flags GJ%FOU, GJ%MSG, and GJ%XTN and clears all other flags. 2. .CMIFI sets flags GJ%OLD, and GJ%XTN and clears all other flags. 3. .GMOFI and .GMIFI zero the right half of word .GJGEN. 4. .CMFIL sets flag GJ%XTN and clears GJ%CFM. |
| .GJSRC | COMND | None |
| .GJDEV - .GJJFN | COMND/ USER | Functions .CMIFI AND |

TOPS-20 MONITOR CALLS
(COMND)

.CMOFI give COMND control of these words. .CMFIL gives the user control of these words.

.GJF2 -
.GJBFP COMND None

.GJATR USER Function .CMFIL gives the user control of this word. .GJATR is not used for other functions.

7 .CMFLD Parse an arbitrary field. This function is useful for fields not normally handled by the COMND call. The input, as delimited by the first nonalphanumeric character, is copied into the atom buffer; the delimiter is not copied. Note the following:

1. This function will parse a null field
2. Hyphens are treated as alphanumeric characters for this function
3. No validation is performed (such as filename validation)
4. No standard help message is available (see description of word .CMHLP, below)
5. The FLDBK. and BRMSK. macros can be used for including other characters in the field (such as the asterisk (*) character)

10 .CMCFM Confirm. This function waits for the user to confirm the command with a carriage return and should be used at the end of parsing a command line.

11 .CMDIR Parse a directory name. Login and files-only directories are allowed. Word .CMDAT contains data bits for this function. The currently defined bit is as follows:

B0(CM%DWC) Allow wildcard characters to be typed in a directory name.

On a successful return, AC2 contains the 36-bit directory number.

TOPS-20 MONITOR CALLS
(COMND)

- 12 .CMUSR Parse a user name. Only login directories are allowed. On a successful return, AC2 contains the 36-bit user number.
- 13 .CMCMA Parse a comma. This function sets B1(CM%NOP-no parse) in word .CMFLG of the command state block and returns an error if a comma is not the next item in the input. Blanks can appear on either side of the comma. This function is useful for parsing a list of arguments.
- 14 .CMINI Initialize the command line by setting up internal monitor pointers, typing the prompt, and checking to see if the user typed CTRL/H. This function should be used before beginning of parsing a command line, but not before reparsing a line. Reinitializing the command line with this function before starting to reparse the command line prevents the use of the CTRL/H feature.

To use this function, the user first moves the needed data into the command state block and then issues .CMINI. If an error occurs while a line is being parsed, .CMINI is issued again by the COMND JSYS to reinitialize the line.

For the second and all subsequent .CMINI function calls for a given line, the user should not alter the byte pointers and character counts in the command state block. To do so would disable the CTRL/H feature. This feature allows the user program, on parsing a bad atom, to print an error message, reissue the prompt, and parse the command line again without forcing the user to retype the entire line.

If .CMINI reads a CTRL/H character, .CMINI resets all byte pointers and character counts except the .CMINC count to their original state. .CMINI sets the .CMINC count to the number of characters in the buffer up to the bad atom. These characters are output to the terminal and parsed again. Control then passes to the reparse address (if provided), and normal parsing resumes. The effect on the program is as if the bad atom had never been typed.

- 15 .CMFLT Parse a floating-point number. On a successful return, AC2 contains the floating-point number.

TOPS-20 MONITOR CALLS
(COMND)

- 16 .CMDEV Parse a device name. A device name consists of up to six alphanumeric characters terminated by a colon (":"). On a successful return, AC2 contains the device designator.
- 17 .CMTXT Parse the input text up to the next carriage return, place the text in the atom buffer, and return. If an ESC or CTRL/F is typed, it causes the terminal bell to ring (because recognition is not available with this function) and is otherwise ignored. If a question mark (?) is typed, an appropriate response is given, and the question mark (?) is not included in the atom buffer. (A question mark can be included in the input text if it is preceded by a CTRL/V. However, if the input text is a user name, the CTRL/V cannot be used to precede a question mark.)
- 20 .CMTAD Parse a date and/or time field according to the setting of bits CM%IDA and CM%ITM. The user must input the field as requested. Any date format allowed by the IDTIM call can be input. If a date is not input, it is assumed to be the current date. If a time is not input, it is assumed to be 00:00:01. When both the date and time fields are input, they must be separated by one or more spaces. If the fields are input separately, they must be terminated with a space or carriage return. Word .CMDAT contains bits in the left half and an address in the right half as data for the function. The bits are:
- B0(CM%IDA) Parse a date
B1(CM%ITM) Parse a time
B2(CM%NCI) Do not convert the date and/or time to internal format. (Refer to Section 2.9.2.)
- The address in the right half is the beginning of a three-word block in the caller's address space. On a successful return, this block contains data returned from the IDTNC call executed by COMND if B2(CM%NCI) was on in the COMND call (if the input date and/or time field was not to be converted to internal format). If B2(CM%NCI) was off in the COMND call, on a successful return, AC2 contains the internal date and time format.
- 21 .CMQST Parse a quoted string up to the terminating quote. The delimiters for the string must be double quotation marks and are not copied to the atom buffer. A double quotation mark is input as part

TOPS-20 MONITOR CALLS
(COMND)

of the string if two double quotation marks appear together. This function is useful if the legal field terminators and the action characters are to be included as part of a string. The characters ?, ESC, and CTRL/F are not treated as action characters, and are included in the string stored in the atom buffer. Carriage return is an invalid character in a quoted string and causes B1(CM%NOP) to be set on return.

- 22 .CMUQS Parse an unquoted string up to one of the specified break characters. Word .CMDAT contains the address of a 4-word block of 128 break character mask bits. (Refer to word .RDBRK of the TEXTI call description for an explanation of the mask.) The characters scanned are not placed in the atom buffer. On return, .CMPTR is pointing to the break character. This function is useful for parsing a string with an arbitrary delimiter. The characters ?, ESC, and CTRL/F are not treated as action characters (unless they are specified in the mask) and can be included in the string. Carriage return can also be included if it is not one of the specified break characters.
- 23 .CMTOK Parse the input and compare it with a given string. Word .CMDAT contains the byte pointer to the given string. This function sets B1(CM%NOP) in word .CMFLG of the command state block and returns if the next input characters do not match the given string. Leading blanks in the input are ignored. This function is useful for parsing single or multiple character operators (for example, + or **).
- 24 .CMNUX Parse a number and terminate on the first nonnumeric character. Word .CMDAT contains the radix (from 2 to 10) of the number. On a successful return, AC2 contains the number. This function is useful for parsing a number that may not be terminated with a nonalphabetic character (for example, 100PRINT FILEA).

Note that nonnumeric identifiers can begin with a digit (for example, 1SMITH as a user name). When a nonnumeric identifier and a number appear as alternates for a field, the order of the function descriptor blocks is important. The .CMNUX function, if given first, would accept the digit in the nonnumeric identifier as a valid number instead of as the beginning character of a nonnumeric identifier.

TOPS-20 MONITOR CALLS
(COMND)

- 25 .CMACT Parse an account string. The input, as delimited by the first nonalphanumeric character, is copied into the atom buffer; the delimiter is not copied. No verification is performed nor is any standard help message available. The length of the string is checked, and if it exceeds 39 characters, an error is generated.
- 26 .CMNOD Parse a network node name. A node name consists of up to six alphanumeric characters followed by 2 colons ("::"). The node name must begin with an alphabetic character. Lowercase characters are converted to uppercase characters. The node name is copied into the atom buffer without the colons.

In addition to the function code in bits 0-8 (CM%FNC), .CMFNP also contains function-specific flag bits in bits 9-17 (CM%FFL), and the address of another function descriptor block in bits 18-35 (CM%LST).

The flag bits that can be set in bits 9-17 (CM%FFL) are as follows:

| Bit | Symbol | Meaning |
|-----|--------|--|
| 11 | CM%NOC | Indicates that a semicolon does not begin a full-line comment and instead is matched with the specified function in the function descriptor block. If this bit is not set, the semicolon begins a full line comment. |
| 12 | CM%NSF | Indicates that a suffix is optional. This bit is meaningful only with the .CMDEV and .CMNOD functions. If this bit is not set, the suffix is required. |
| 13 | CM%BRK | Notifies COMND that word .CMBRK of the function descriptor block contains a pointer to a 4-word break mask. See description of word .CMBRK for more details. |
| 14 | CM%PO | The field is to be parsed only, and the field's existence is not to be verified. This bit currently applies to the .CMDEV, .CMDIR, .CMNOD, and .CMUSR functions and is ignored for the remaining functions. On return, COMND sets B1(CM%NOP-no parse) only if the field typed is not in the correct syntax. Also, data returned in AC2 may not be correct. |
| 15 | CM%HPP | A byte pointer to a program-supplied help message for this field is given in word 2 (.CMHLP) of this function descriptor block. |

TOPS-20 MONITOR CALLS
(COMND)

- 16 CM%DPP A byte pointer to a program-supplied default string for this field is given in word 3 (.CMDEF) of this function descriptor block.
- 17 CM%SDH The output of the default help message is to be suppressed if the user types a question mark. (See below for the default messages.)

The address of another function descriptor block can be given in bits 18-35 (CM%LST) of the .CMFNP word. The use of this second descriptor block is described below.

Usually one COMND call is executed for each field in the command. However, for some fields, more than one type of input may be possible (for example, after a keyword field, the next field could be a switch or a filename field). In these cases, all the possibilities for a field must be tried in an order selected to test unambiguous cases first.

When the COMND call cannot parse the field as indicated by the function code, it does one of two things:

1. It sets the current pointer and counts such that the next call will attempt to parse the same input over again. It then returns with B1(CM%NOP) set in the left half of the .CMFLG word in the command state block. The caller can then issue another COMND call with a function code indicating another of the possible fields. After the execution of each call, the caller should test the CM%NOP flag to see that the field was parsed successfully.
2. If an address of another function descriptor block is given in CM%LST, the COMND call moves to this descriptor block automatically and attempts to parse the field as indicated by the function code contained in B0-8(CM%FNC) in word .CMFNP of that block. If the COMND call fails to parse the field using this new function code, it moves to a third descriptor block if one is given. This sequence continues until either the field is successfully parsed or the end of the chain of function blocks is reached. Upon completion of the COMND call, AC3 contains the addresses of the first and last function blocks used.

By specifying a chained list of function blocks, the program can have the COMND call automatically check all possible alternatives for a field and not have to issue a separate call for each one. In addition, if the user types a question mark, a list is output of all the alternatives for the field as indicated by the list of function descriptor blocks.

TOPS-20 MONITOR CALLS
(COMND)

Word .CMHLP of the Function Descriptor Block

This word contains a byte pointer to a program-supplied help text. The COMND call outputs this help if the user types a question mark when entering a command field. Bit 15(CM%HPP) must be set in word 0 (.CMFNP) of the function descriptor block for this pointer to be used.

If B17(CM%SDH) is set in this word, COMND outputs only the program-supplied message. If B17(CM%SDH) is not set, COMND appends the default help message to the program-supplied message, and outputs them both.

If .CMHLP is zero, COMND outputs only the default message.

The default help message depends on the particular function being used to parse the current field. The following table lists the default help message for each function available in the COMND call.

Default Help Messages

| Function | Message |
|---------------------|--|
| .CMKEY (keyword) | One of the following followed by the alphabetical list of valid keywords. If the user types a question mark in the middle of the field, only the keywords that can possibly match the field as currently typed are output. If no keyword can possibly match the currently typed field, the following message is output: keyword (no defined keywords match this input). If there is only 1 keyword, the keyword becomes the HELP message. |
| .CMNUM (number) | The help message output depends on the radix specified in .CMDAT in the descriptor block. If the radix is octal, the help message is octal number. If the radix is decimal, the help message is decimal number. If the radix is any other radix, the help message is a number in base <u>nn</u> where <u>nn</u> is the radix. |
| .CMNOI (guide word) | None |
| .CMSWI (switch) | One of the following followed by the alphabetical list of valid switch keywords. The same rules apply as for .CMKEY function, above. |

TOPS-20 MONITOR CALLS
(COMND)

| | |
|-------------------------|---|
| .CMIFI (input file) | The help message output depends on the settings of certain bits in the GTJFN call. |
| .CMOFI (output file) | If bit GJ%OLD is off and bit GJ%FOU is on, the help message is output filespec. |
| .CMFIL (any file) | Otherwise, the help message is input filespec. |
| | |
| .CMFLD (any field) | None |
| .CMCFM (confirm) | Confirm with carriage return |
| .CMDIR (directory) | Directory name |
| .CMUSR (user) | User name |
| .CMCMA (comma) | Comma |
| .CMINI (initialize) | None |
| .CMFLT (floating point) | Number |
| .CMDEV (device) | Device name |
| .CMTXT (text) | Text string |
| .CMTAD (date) | The help message depends on the bits set in .CMDAT in the descriptor block. If CM%IDA is set, the help message is date. If CM%ITM is set, the help message is time. If both are set, the help message is date and time. |
| .CMQST (quoted) | Quoted string |
| .CMUQS (unquoted) | Unquoted string if "?" is a break character, otherwise none |
| .CMTOK (token) | None |
| .CMNUX (number) | Same as .CMNUM |
| .CMACT (account) | None |
| .CMNOD (node) | Node name |

Word .CMDEF of the Function Descriptor Block

This word contains a byte pointer to the ASCIZ string to be used as the default for this field. For this pointer to be used, bit 16 (CM%DPP) must be set in word 0 (.CMFNP) of the descriptor block. The string is output to the destination, as well as copied to the text

TOPS-20 MONITOR CALLS
(COMND)

buffer, if the user types an ESC or CTRL/F as the first nonblank character in the field. If the user types a carriage return, the string is copied to the atom buffer, but is not output to the destination.

When the caller supplies a list of function descriptor blocks, the byte pointer for the default string must be included in the first block. The CM%DPP bit and the pointer for the default string are ignored when they appear in subsequent blocks. However, the default string can be worded so that it applies to any of the alternative fields. The effect is the same as if the user had typed the given string.

Defaults for fields of a file specification can also be supplied with the .CMFIL function. If both the byte pointer to the default string and the JFN defaults have been provided, the COMND default is used first, and then, if necessary, the GTJFN defaults are used.

NOTE

The function descriptor block, whose address is given in AC2, can be set up by the FLDDDB. and FLDBK. macros defined in MACSYM. (See the end of the COMND section for a description of these macros.)

Word .CMBRK of the Function Descriptor Block

This word contains a pointer to a 4-word user-specified mask that determines which characters constitute end of field. The leftmost 32 bits of each word correspond to a character in the ASCII collating sequence (in ascending order). If the bit is on for a given character, typing that character causes the COMND JSYS to treat the characters typed so far as a separate field and to parse them according to the function being used. CM%BRK (B13) must be on in the first word of the function descriptor block, or COMND ignores word .CMBRK.

Ordinarily, the user relies on COMND's default masks (varying according to function) to specify which characters signal end of field, and thus is not concerned with word .CMBRK of the function block. But for special purposes such as allowing "*" or "%" to be part of a field, rather than a field delimiter, the user must specify his own mask. (In this example, the bits for "*" and "%" would be off in the mask word.) The user may inspect COMND's default masks (defined in MONSYM) for help in designing a custom mask.

The following is a list of the COMND functions that use masks:

TOPS-20 MONITOR CALLS
(COMND)

| Mask Symbols | COMND Function | Changeable by User |
|-----------------|----------------|--------------------------|
| KEYB0. - KEYB3. | .CMKEY | Yes |
| DEVB0. - DEVB3. | .CMDEV | Yes (only if parse-only) |
| FLDB0. - FLDB3. | .CMFLD | Yes |
| EOLB0. - EOLB3. | .CMTXT | Yes |
| KEYB0. - KEYB3. | .CMSWI | Yes |
| User-specified | .CMTAD | Yes |
| USRB0. - USRB3. | .CMUSR | No |
| FILB0. - FILB3. | .CMFIL | No |
| FILB0. - FILB3. | .CMIFI | No |
| FILB0. - FILB3. | .CMOFI | No |
| internal | .CMNUM | No |
| FILB0. - FILB3. | .CMDIR | No |
| internal | .CMFLT | No |
| ACTB0. - ACTB3. | .CMACT | No |

COMND will ignore any break masks that are specified for functions that do not allow user-modified masks.

Note that specifying a zero mask with CM%BRK set will cause the TTY line buffer to fill up and generate an error.

On a successful return, the COMND call returns flag bits in the left half of AC1 and preserves the address of the command state block in the right half of AC1. These flag bits are copied from word .CMFLG in the command state block and are described as follows.

Bits Returned on COMND Call

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | CM%ESC | An ESC was typed by the user as the terminator for this field. |
| 1 | CM%NOP | The field could not be parsed because it did not conform to the specified function(s). An error code is returned in AC2. If this bit is set, bits 0 (CM%ESC) and 2 (CM%EOC) might not contain valid information. |
| 2 | CM%EOC | The field was terminated with a carriage return. |
| 3 | CM%RPT | Characters already parsed need to be reparsed because the user edited them. This bit does not need to be examined if the program has supplied a reparse dispatch address in the right half of .CMFLG in the command state block. |

TOPS-20 MONITOR CALLS
(COMND)

- 4 CM%SWT A switch field was terminated with a colon. This bit is on if the user either used recognition on a switch that ends with a colon or typed a colon at the end of the switch.
- 5 CM%PFE The previous field was terminated with an ESC.

When a field cannot be parsed, B1(CM%NOP) is set in AC1, and an error code is returned in AC2. Note that if a list of function descriptor blocks is given and an error code is returned, the error is associated with the function that had the largest atom buffer after all function blocks have been tried without a successful parse of the field.

- NPXAMB: Ambiguous
NPXNSW: Not a switch - does not begin with slash
NPXNOM: Does not match switch or keyword
NPXNUL: Null switch or keyword given
NPXINW: Invalid guide word
NPXNC: Not confirmed
NPXICN: Invalid character in number
NPXIDT: Invalid device terminator
NPXNQS: Not a quoted string - does not begin with double quote
NPXNMT: Does not match token
NPXNMD: Does not match directory or user name, or structure not mounted
NPXCMA: Comma not given
COMX18: Invalid character in node name
COMX19: Too many characters in node name

Macros

Several macros (defined in MACSYM) are available to make using the COMND JSYS more convenient. These macros are as follows:

FLDDB.(TYP,FLGS,DATA,HLPM,DEFM,LST)

where:

- TYP = function type
FLGS = function flags
DATA = function-specific data
HLPM = help message
DEFM = default text
LST = additional invocations of the FLDDB. macro (used only if multiple function blocks are required)

This macro generates function descriptor blocks for COMND. For example, the following code performs a .CMINI function:

```
MOVEI T1,STEBLK                   ;Get address of COMND state block  
MOVEI T2,[FLDDB.(.CMINI)]       ;Get address of function block
```

TOPS-20 MONITOR CALLS
(COMND)

COMND

The following code performs a .CMKEY function (assuming that the keyword table started at address CMDTAB:

```
MOVEI T1,STEBLK           ;Get address of COMND state block
MOVEI T2,[FLDDB(.CMKEY,<CM%DPP+CM%HPP>,CMDTAB,
              <help text>,<default text>)]
COMND
```

FLDBK.(TYP,FLGS,DATA,HLPM,DEFM,BRKADR,LST)

This is exactly the same as FLDDB., except that a provision has been made for the address of the first word of a 4-word character mask (BRKADR). This version is for use when a user-specified character mask is required.

BRMSK.(INI0,INI1,INI2,INI3,ALLOW,DISALLOW)

where:

```
INI0 = first word of character mask
INI1 = second word of character mask
INI2 = third word of character mask
INI3 = fourth word of character mask
ALLOW = characters to allow in the mask
DISALLOW = characters to disallow in the mask
```

This macro generates 4-word character masks for use with those COMND functions that allow the user to specify his own mask. For example, executing the following code allows "*" in the predefined mask for the .CMFLD function (FLDB0 thru FLDB3):

```
BRMSK.(FLDB0.,FLDB1.,FLDB2.,FLDB3.,<*>,)
```

Also, the BRMSK. macro may be invoked within the FLDBK. macro:

```
FLDBK.(TYP,FLGS,DATA,HLPM,DEFM,[
      BRMSK.(INI0,INI1,INI2,INI3,ALLOW,DISALLOW)],LST)
```

The COMND call causes other monitor calls to be executed, depending on the particular function that is requested. Failure of these calls usually results in the failure to parse the requested field. In these cases, the relevant error code can be obtained by the GETER and ERSTR monitor calls.

Any TBLUK error can occur on the keyword and switch functions.

Any NIN/NOU and FLIN/FLOUT error can occur on the number functions.

TOPS-20 MONITOR CALLS
(COMND)

Any GTJFN error except for GJFX37 can occur on the file specification functions.

Any IDTNC error can occur on the date/time function.

Any RCDIR or RCUSR error can occur on the directory and user functions.

Any STDEV error can occur on the device function.

Generates an illegal instruction interrupt on error conditions below.

COMND ERROR MNEMONICS:

COMNX1: Invalid COMND function code
COMNX2: Field too long for internal buffer
COMNX3: Command too long for internal buffer
COMNX5: Invalid string pointer argument
COMNX8: Number base out of range 2-10
COMNX9: End of input file reached
COMX10: Invalid default string
COMX11: Invalid CMRTY pointer
COMX12: Invalid CMBFP pointer
COMX13: Invalid CMPTR pointer
COMX14: Invalid CMABP pointer
COMX15: Invalid default string pointer
COMX16: Invalid help message pointer
COMX17: Invalid byte pointer in function block
VACCX1: Account string too long

Creates, changes, or deletes a directory entry.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Byte pointer to ASCIZ string containing the structure and directory name. The string must be of the form: structure:<directory>.

AC2: B0(CD%LEN) Set flags and length of the argument block from the values given in word .CDLEN.

B1(CD%PSW) Set password from argument block

B2(CD%LIQ) Set working disk storage limit from argument block

TOPS-20 MONITOR CALLS
(CRDIR)

B3(CD%PRV) Set capability bits from argument block

B4(CD%MOD) Set mode bits from argument block

B5(CD%LOQ) Set permanent disk storage limit from argument block

B6(CD%NUM) Set directory number from argument block (valid only when creating a directory)

B7(CD%FPT) Set default file protection from argument block

B8(CD%DPT) Set directory protection from argument block

B9(CD%RET) Set default retention count from argument block

B10(CD%LLD) Set last LOGIN date from argument block

B11(CD%UGP) Set user groups from argument block

B12(CD%DGP) Set directory groups from argument block

B13(CD%SDQ) Set subdirectory quota from argument block

B14(CD%CUG) Set user groups assignable by this directory from argument block

B15(CD%DAC) Set default account from argument block

B16(CD%PPN) Set project-programmer number from argument block

B17(CD%DEL) Delete this directory entry

B18-35(CD%APB) Address of the argument block

AC3: Byte pointer to ASCIZ string containing the password of the directory. This pointer is required when a nonprivileged user is changing parameters for his directory.

RETURNS +1: Always, with directory number in AC1

This monitor call requires the process to have WHEEL or OPERATOR capability enabled unless one of the following conditions is true:

TOPS-20 MONITOR CALLS
(CRDIR)

1. The specified directory is one to which the caller has owner access, and the caller is changing any one of the following parameters:

password (.CDPSW)
 default file protection (.CDFPT)
 directory protection (.CDDPT)
 default retention count (.CDRET)
 default account (.CDDAC)

This feature is installation dependent and is enabled by issuing function .SFCRD of the SMON monitor call.

2. The specified directory is inferior to the one to which the caller is currently connected, and the caller has owner access to this inferior directory.

Refer to Section 2.2.6 for the description of owner access.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|------------|---|
| 0 | .CDLEN | Flag bits in the left half, and length of the argument block in the right half. The following bits are defined: |
| | B0(CD%NSQ) | When restoring this directory, do not update its superior directory's quotas (permanent, working, and subdirectory quotas) to account for this directory. If this bit is off, the superior directory's quotas are updated. This bit is set by the DLUSER or DUMPER program to retain the superior directory's quotas when restoring its subdirectories. The process must have WHEEL or OPERATOR capability enabled to set this bit. |
| | B1(CD%NCE) | When restoring or reconstructing this directory, do not change any directory parameters if the directory currently exists on disk; set the parameters only if the directory does not exist. If this bit is off, the directory parameters as saved are restored for the directory. This bit is set by the DLUSER or DUMPER program to restore or reconstruct directories from out-of-date files without causing |

TOPS-20 MONITOR CALLS
(CRDIR)

existing directories to revert to older parameters. The process must have WHEEL or OPERATOR capability enabled to set this bit.

B2(CD%NED) Set default on-line expiration date from word .CDDNE.

B3(CD%FED) Set default off-line expiration date from word .CDDFE.

B4(CD%RNA) Reserved for DIGITAL.

B5(CD%PEN) Set password encryption version from word .CDPEV and encryption date from word .CDPDT.

B6(CD%PED) Set password expiration date from word .CDPED.

B7(CD%PMU) Set maximum password use count from .CDPMU.

B8(CD%SNI) Set last non-interactive login date and time from argument block.

B9(CD%SFC) Set number of failed logins (interactive and non-interactive) from argument block.

1 .CDPSW Byte pointer to password string, which is a string from 1 to 39 alphanumeric characters (including hyphens).

2 .CDLIQ Maximum number of pages that can be used for working disk storage (also known as logged-in quota).

3 .CDPRV Capabilities for this user. (Refer to Section 2.7.1 for the capability bits.)

4 .CDMOD Mode word.

B0(CD%DIR) Directory name can be used only to connect to (the directory is a files-only directory). If this bit is off, the directory name can be used for logging in and connecting to.

B1(CD%ANA) Accounts are alphanumeric. This bit is not used and is provided for

TOPS-20 MONITOR CALLS
(CRDIR)

compatibility with systems earlier than TOPS-20 version 3.

- B2(CD%RLM) All messages from the file <SYSTEM>MAIL.TXT are repeated each time the user logs in. If this bit is off, only the messages not previously printed are output when the user logs in.
- B7(CD%DAR) If on, this bit indicates that the file should be archived rather than migrated to virtual disk when the on-line expiration date has been reached.
- B8(CD%SEC) If on, files created are set secure by default.
- 5 .CDLOQ Maximum number of pages that can be used for permanent disk storage (also known as logged-out quota).
- 6 .CDNUM Directory number, valid only when creating a directory. An error code is returned if the user changes the number of an existing directory (CRDIX2) or gives a nonunique number (CRDIX8).
- 7 .CDFPT Default file protection (18 bits, right-justified).
- 10 .CDDPT Directory protection (18 bits, right-justified).
- 11 .CDRET Default number of generations of a file to be retained in the directory (retention count). Valid numbers are 0 to 63, with 0 being an infinite number.
- 12 .CDLLD Date and time of last interactive login.
- 13 .CDUGP Address of user group list for this directory.
- 14 .CDDGP Address of directory group list.
- 15 .CSDSQ Maximum number of directories that can be created inferior to this directory. This parameter allows a user to create directories with the BUILD command.
- 16 .CDCUG Address of user group list. This list contains the group numbers that can be assigned to subdirectories.

TOPS-20 MONITOR CALLS
(CRDIR)

| | | |
|----|--------|--|
| 17 | .CDDAC | Byte pointer to default account string for this user. |
| 20 | .CDDNE | Default on-line expiration date and time, which can be an explicit date and time (internal format) or an interval (in days). In either case, the specified date/interval cannot exceed the system maximum. This parameter is read if CD%NED (1B2) or CD%FED (1B3) in .CDLEN are set. If a new directory is created and this parameter is not specified, the system default is used. An unprivileged user can modify his defaults to be less than or equal to those that are currently specified or the system maximum, whichever is greater. A user with WHEEL capability may override the system maximum. If no system maximum has been specified, there is no on-line expiration date and time associated with the directory. |
| 21 | .CDDFE | Default off-line expunge date and time. Otherwise similar to .CDDNE (above). |
| 22 | .CDDRN | Reserved for DIGITAL. |
| 23 | .CDPEV | Version number of password encryption algorithm. |
| 24 | .CDPDT | Date password was encrypted. |
| 25 | .CDPED | Date password expires. |
| 26 | .CDPMU | Maximum use count for password. |
| 27 | .CDPPN | TOPS-10 Project-Programmer number: p,,pn requires WHEEL or OPERATOR capability to set project number (p) less than 10; project number cannot be 4. |
| 30 | .CDNLD | Date and time of last non-interactive login. |
| 31 | .CDFPA | Count of failed interactive logins for this user in the left half,,count of failed non-interactive logins in the right half. |

The format of each group list is a table with the first word containing a count of the number of words (including the count word) in the table and each subsequent word containing a group number.

When CRDIR is being executed to create a directory, bits 0-17 of AC2 can optionally be on or off. If a particular bit is on, it indicates that the corresponding argument in the argument block should be

TOPS-20 MONITOR CALLS
(CRDIR)

examined. If the bit is off, it indicates that the argument should be defaulted.

The following lists the bits and the corresponding argument defaults:

| Bits | Argument Defaults |
|-------------|--|
| B2(CD%LIQ) | Maximum working disk file storage to 250 pages |
| B3(CD%PRV) | No special capabilities |
| B4(CD%MOD) | Directory name that can be used for logging in and that lists the messages from <SYSTEM>MAIL.TXT only once |
| B5(CD%LOQ) | Maximum permanent disk file storage to 250 pages |
| B6(CD%NUM) | The first unused directory number; B6 should normally be off. |
| B7(CD%FPT) | Default file protection to 777700 |
| B8(CD%DPT) | Directory protection to 777700 |
| B9(CD%RET) | Default file retention count to 1 |
| B10(CD%LLD) | Never logged in |
| B11(CD%UGP) | No user groups |
| B12(CD%DGP) | No directory groups |
| B13(CD%SDQ) | No ability to create inferior directories |
| B14(CD%CUG) | No assignable user groups for inferior directories |
| B15(CD%DAC) | No default account |

When CRDIR is being executed to change a directory and any of B0-17 of AC2 is off, the corresponding parameter is not affected.

When CRDIR is being executed to delete a directory, the settings of B0-17 of AC2 are ignored. A CRDIR call cannot be given to delete a directory that has directories inferior to it.

The GTDIR call can be used to obtain the directory information.

Generates an illegal instruction interrupt on error conditions below.

CRDIR ERROR MNEMONICS:

| | |
|---------|--|
| ACESX3: | Password required |
| CRDIX1: | WHEEL or OPERATOR capability required |
| CRDIX2: | Illegal to change number of old directory |
| CRDIX3: | Insufficient system resources (Job Storage Block full) |
| CRDIX4: | Superior directory full |
| CRDIX5: | Directory name not given |
| CRDIX6: | Directory file is mapped |
| CRDIX7: | File(s) open in directory |
| CRDIX8: | Invalid directory number |
| CRDIX9: | Internal format of directory is incorrect |
| CRDI10: | Maximum directory number exceeded; index table needs expanding |

TOPS-20 MONITOR CALLS
(CRDIR)

CRDI11: Invalid terminating bracket on directory
CRDI12: Structure is not mounted
CRDI13: Request exceeds superior directory working quota
CRDI14: Request exceeds superior directory permanent quota
CRDI15: Request exceeds superior directory subdirectory quota
CRDI16: Invalid user group
CRDI17: Illegal to create nonfiles-only subdirectory under
files-only directory
CRDI18: Illegal to delete logged-in directory
CRDI19: Illegal to delete connected directory
CRDI20: WHEEL, OPERATOR, or requested capability required
CRDI21: Working space insufficient for current allocation
CRDI22: Subdirectory quota insufficient for existing subdirectories
CRDI23: Superior directory does not exist
CRDI24: Invalid subdirectory quota
CRDI29: Illegal to disallow subdirectory user group while in use
CRDI30: Invalid password length
CRDI31: Password expiration date is too far in the future
CRDI32: Password expiration is not enabled on this system
CRDI33: Password found in system password dictionary.
ENACX5: Account validation data base file is empty
STRX10: Structure is offline

Creates a new job and optionally logs it in. This monitor call causes the functions that are normally performed when a job is created (for example, assignment of a JSB, the primary I/O designators, and the job controlling terminal) to be performed for the new job.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Flag bits,,0

AC2: Address of argument block

AC3: (optional) If CRJOB is to be used to release control over a job previously created with CRJOB (bit 17 in AC1 must be on), then AC3 contains the job number of the previously-created job.

RETURNS +1: Failure, with error code in AC1

+2: Success, with the number of the new job in AC1

TOPS-20 MONITOR CALLS
(CRJOB)

The flag bits defined in the left half of AC1 are as follows:

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | CJ%LOG | Log in the new job. If this bit is off, the new job is created but not logged in. |
| 1 | CJ%NAM | Set the user name and password from the argument block. If this bit is off, the user name of the caller is given to the new job. |
| 2-3 | CJ%ACT | Set the account of the new job to the following: |
| | | Code Symbol Meaning |
| | | 0 .CJUCA Use current account of caller. |
| | | 1 .CJUAA Use account from the argument block. |
| | | 2 .CJUDA Use default account of user whose job is being created. |
| 4 | CJ%ETF | If set, place the TOPS-20 command processor in the top-level process of the new job. The command processor reads its program argument block (see below) at the time it is started. |

CJ%FIL and CJ%ETF interact in the following ways:

1. If CJ%FIL is on and CJ%ETF is on, then a job is created with a top process consisting of the TOPS-20 command processor and an inferior process consisting of the file to which word .CJFIL points.
2. If CJ%FIL is off and CJ%ETF is on, then a job is created with a top process consisting of the TOPS-20 command processor. No inferior process is created.
3. If CJ%FIL is on and CJ%ETF is off, then a job is created with a top process consisting of the file to which word .CJFIL points. No inferior process is created.

The format of the program argument block is as follows:

TOPS-20 MONITOR CALLS
(CRJOB)

| Word | Contents |
|------|---|
| 0 | Count of words in block, not including this word. |
| 1 | 1B0+3B6+2B12+CR%PRA - indicates this is a program argument block created by the CRJOB JSYS. |
| 2 | 1B0 + offset1 - offset1 is the offset in this block of the first argument being passed. |
| 3 | 1B0 + offset2 - offset2 is the offset in this block of the second argument being passed. |
| n | (offset1) This argument is a copy of the flag bits from word 10 (.CJEXF) of the CRJOB argument block, which contains the flags for the command language processor. |
| n+1 | (offset2) This argument contains information about the process being started: the process handle in the left half, and the entry vector offset in the right half. The entry vector offset is from word .CJSVF (word 4) of the CRJOB argument block. |

The program argument block is created by the CRJOB monitor call and is passed to the process by a PRARG monitor call (performed internally by CRJOB). The user does not specify any of the information in the program argument block. Only the program at the top fork level of the job (usually the TOPS-20 EXEC) can read the PRARG block.

| | | |
|---|--------|--|
| 5 | CJ%FIL | Move the file to which a word in the argument block points into a process in the new job (by means of a GET call). If B4(CJ%ETF) is off, the file is placed in the top-level process of the new job. If B4(CJ%ETF) is on, the file is placed in the process designated in the Command Language Processor's PRARG argument block (see below). |
|---|--------|--|

If B5(CJ%FIL) is off, no file is moved into a process of the new job, and the top-level process of the new job is the Command Language Processor.

TOPS-20 MONITOR CALLS
(CRJOB)

- | | | |
|----|--------|---|
| 6 | CJ%ACS | Load the ACs from the value in the argument block. The ACs are loaded only if a program other than the Command Language Processor is being run. |
| 7 | CJ%OWN | Maintain ownership of the new job. This means that when the caller logs out, the new job is also logged out. However, the new job can also be logged out by the normal mechanisms. If this bit is off, control of the new job is released. |
| 8 | CJ%WTA | Do not start the new job until it is attached (using ATACH JSYS) to a terminal. If this bit is off, the new job is started. |
| 9 | CJ%NPW | Do not check the password given when the new job is logged in. If this bit is off, the password is checked unless the new job is being logged in with the same user name as the caller, or with WHEEL or OPERATOR capability enabled. |
| 10 | CJ%NUD | Do not update the date of LOGIN for the user logging in to the new job. If this bit is off, the date of LOGIN is updated, unless the user is logging in with the same user name as the caller, or with WHEEL or OPERATOR capability enabled. |
| 11 | CJ%SPJ | Set (by means of a SPJFN call) the primary input and output designators from the argument block before starting the job. The primary I/O designators are not changed for a Command Language Processor in the top-level process of the new job; they are changed only for inferior processes. If this bit is off, the primary I/O designators of the new job are the job's controlling terminal. |
| 12 | CJ%CAP | Set the allowed user capabilities of the new job (right half) to be the same as the caller's currently enabled capabilities, until the new job is logged in. If this bit is off, the new job has the user capabilities associated with the user whose job is being created. |
| 13 | CJ%CAM | Set the allowed user capabilities of the new job to the combination of (AND function) the capability mask in the argument block and the new job's user capabilities. If this bit is off, the new job has the capabilities associated with the user whose job is being created. |
| 14 | CJ%SLO | Send an IPCF message to the PID supplied in the argument block when the new job is logged out. If |

TOPS-20 MONITOR CALLS
(CRJOB)

this bit is off, no message is sent when the new job is logged out.

The IPCF logout message has the following format:

| Word | Contents | | | | | | | | | | | | | | | | | | |
|------|---|--|--------|---------|----|--------|-----------------------------|----|--------|-----------------------|----|--------|-------------------------|----|--------|--|----|--------|---|
| 0 | 0,,.IPCL0 | | | | | | | | | | | | | | | | | | |
| 1 | N,,# of job logged out. N is the count of the remaining words in this message (currently 10 octal). | | | | | | | | | | | | | | | | | | |
| 2 | flags,,reserved | | | | | | | | | | | | | | | | | | |
| | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Bits</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>B0</td> <td>SP%BAT</td> <td>job is controlled by batch.</td> </tr> <tr> <td>B1</td> <td>SP%DFS</td> <td>spooling is deferred.</td> </tr> <tr> <td>B2</td> <td>SP%ELO</td> <td>the job executed LGOUT.</td> </tr> <tr> <td>B3</td> <td>SP%FLO</td> <td>the job was forced to logout. If this bit is on, check word 10 of the IPCF message (gives code of most recent monitor call error). B3 will be on only if the job has an interrupt to be handled by MEEXEC (Mini-EXEC).</td> </tr> <tr> <td>B4</td> <td>SP%OLO</td> <td>the job was logged out by another job. Word 6 of the IPCF message contains the job number of the job that did the logout.</td> </tr> </tbody> </table> | Bits | Symbol | Meaning | B0 | SP%BAT | job is controlled by batch. | B1 | SP%DFS | spooling is deferred. | B2 | SP%ELO | the job executed LGOUT. | B3 | SP%FLO | the job was forced to logout. If this bit is on, check word 10 of the IPCF message (gives code of most recent monitor call error). B3 will be on only if the job has an interrupt to be handled by MEEXEC (Mini-EXEC). | B4 | SP%OLO | the job was logged out by another job. Word 6 of the IPCF message contains the job number of the job that did the logout. |
| Bits | Symbol | Meaning | | | | | | | | | | | | | | | | | |
| B0 | SP%BAT | job is controlled by batch. | | | | | | | | | | | | | | | | | |
| B1 | SP%DFS | spooling is deferred. | | | | | | | | | | | | | | | | | |
| B2 | SP%ELO | the job executed LGOUT. | | | | | | | | | | | | | | | | | |
| B3 | SP%FLO | the job was forced to logout. If this bit is on, check word 10 of the IPCF message (gives code of most recent monitor call error). B3 will be on only if the job has an interrupt to be handled by MEEXEC (Mini-EXEC). | | | | | | | | | | | | | | | | | |
| B4 | SP%OLO | the job was logged out by another job. Word 6 of the IPCF message contains the job number of the job that did the logout. | | | | | | | | | | | | | | | | | |
| 3 | job connect time | | | | | | | | | | | | | | | | | | |
| 4 | job CPU time | | | | | | | | | | | | | | | | | | |
| 5 | TTY number of job at logout (-1 if detached) | | | | | | | | | | | | | | | | | | |
| 6 | job number of the job that did the logout | | | | | | | | | | | | | | | | | | |
| 7 | reserved | | | | | | | | | | | | | | | | | | |
| 10 | code of the most recent monitor call error | | | | | | | | | | | | | | | | | | |

17 CJ%DSN Release ownership of the previously created job whose number is in AC3. If this bit is on, it overrides the setting of all other bits in AC1; and no change is made to the job's status other than the change in ownership.

TOPS-20 MONITOR CALLS
(CRJOB)

The format of the argument block (whose address is given in AC2) is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .CJNAM | Byte pointer to the user name string. |
| 1 | .CJPSW | Byte pointer to the password string. |
| 2 | .CJACT | 5B2 + numeric account number or byte pointer to account string. |
| 3 | .CJFIL | Byte pointer to the name of the file to be moved (by a GET call) into a process of the new job. The new job must have read access to the file. The process into which the file is placed depends on the setting of B4(CJ%ETF). |
| 4 | .CJSFV | Offset in the entry vector to use as the start address of the file to which word .CJFIL points. This offset is the argument to the SFRKV call used to start the process. |
| 5 | .CJTTY | Terminal designator of the new job's controlling terminal. This terminal must be assigned by the caller. The terminal is then released and assigned to the new job. If the new job is to be detached, the .NULIO designator (377777) is given. |
| 6 | .CJTIM | Connect-time for new job before a LGOUT is forced on it; 0 indicates no limit. |
| 7 | .CJACS | Address of a 16-word block whose contents are to be loaded in the new job's ACs if a program other than the Command Language Processor is being run. |
| 10 | .CJEXF | Flag bits to be passed to the Command Language Processor in the top-level process of the new job. The bits are: <ul style="list-style-type: none"> B0 Suppress the herald printed by the Command Language Processor. B1 Move the file to which word .CJFIL points into the process whose handle is in the PRARG block (see below). B2 Start the process at the offset in the entry vector given in word .CJSFV. This |

TOPS-20 MONITOR CALLS
(CRJOB)

process is started after the Command Language Processor is initialized.

B3 Output the text printed when a LOGIN command is given (system messages, job number, or terminal number, for example).

This word is copied into the PRARG argument block passed to the Command Language Processor (see below).

- | | | |
|----|--------|---|
| 11 | .CJPRI | Primary input and output designators for the inferior processes of the new job. These designators must refer to device designators. The Command Language Processor in the top-level process of the new job executes an SPJFN call to set these designators. |
| 12 | .CFCPU | Run-time limit for the new job. When this limit is reached, an interrupt is generated (by a TIMER call), and the Command Language Processor executes a LGOUT call for the new job. A zero in this word means there is no run-time limit on the job. |
| 13 | .CJCAM | Capability mask for the new job. This mask is used only if CJ%CAM is set. |
| 14 | .CJSLO | PID to which an IPCF message is to be sent when the new job is logged out. |

When CRJOB creates a new job, it also creates the top-level process, which is always a virgin process. Thus, an execute-only program can be run as the top-level fork.

The CRJOB call causes other monitor calls to be executed, depending on the particular function that is performed.

Any GTJFN and OPENF errors can occur when obtaining the specified file.

Any SFRKV error can occur when starting the program in the specified file.

Any LOGIN and account validation errors can occur when logging in the job.

CRJOB ERROR MNEMONICS:

CRJBX1: Invalid parameter or function bit combination

TOPS-20 MONITOR CALLS
(CRJOB)

CRJBX2: Illegal for created job to enter MINI-EXEC
CRJBX4: Terminal is not available
CRJBX5: Unknown name for LOGIN
CRJBX6: Insufficient system resources

Defines or deletes a logical name assignment. Logical names are used to specify a set of default values for each field requested by a GTJFN monitor call. When a logical name is passed to the GTJFN call, any fields not specified by the user are supplied from the fields defined in the logical name definition. (See Section 2.2.2 and to the INLNM and LNMST monitor call descriptions for more information on logical names.)

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code

AC2: Byte pointer to the logical name (No terminating colon should be supplied.)

AC3: Byte pointer to the logical name definition string

RETURNS +1: Failure, error code in AC1

+2: Success, updated string pointer in AC3

The codes for the functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .CLNJ1 | Delete one logical name from the job |
| 1 | .CLNS1 | Delete one logical name from the system (WHEEL or OPERATOR capability required) |
| 2 | .CLNJA | Delete all logical names from the job |
| 3 | .CLNSA | Delete all logical names from the system (WHEEL or OPERATOR capability required) |
| 4 | .CLNJB | Create a logical name for the job |
| 5 | .CLNSY | Create a logical name for the system (WHEEL or OPERATOR capability required) |

TOPS-20 MONITOR CALLS
(CRLNM)

CRLNM ERROR MNEMONICS:

ARGX09: Invalid byte size
CRLNX1: Logical name is not defined
CRLNX2: WHEEL or OPERATOR capability required
CRLNX3: Invalid function
GJFX4: Invalid character in file name
GJFX5: Field cannot be longer than 39 characters
GJFX6: Device field not in a valid position
GJFX7: Directory field not in a valid position
GJFX8: Directory terminating delimiter is not preceded by a valid beginning delimiter
GJFX9: More than one name field is not allowed
GJFX10: Generation number is not numeric
GJFX11: More than one generation number field is not allowed
GJFX12: More than one account field is not allowed
GJFX13: More than one protection field is not allowed
GJFX14: Invalid protection
GJFX15: Invalid confirmation character
GJFX22: Insufficient system resources (Job Storage Block full)
GJFX31: Invalid wildcard designator

Dismisses the software interrupt routine in progress and resumes the process at the location specified by the PC stored in the priority level table. (See Section 2.6.7.)

RETURNS +1: Only if no software interrupt is currently in progress and if an ERJMP or ERCAL instruction follows the DEBRK

Generates an illegal instruction interrupt on error conditions below.

DEBRK ERROR MNEMONICS:

DBRX1: No interrupts in progress

Reclaims disk space by expunging disk files that have been marked for deletion with DELF. This call first checks to see that the user has connect access to the directory. The calling process must have connect access to the directory to expunge files from it.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

TOPS-20 MONITOR CALLS
(DELDF)

ACCEPTS IN AC1: B0(DD%DTF) Delete temporary files (;T) also

B1(DD%DNF) Delete nonexistent files that are not now open

B2(DD%RST) Rebuild the symbol table

B3(DD%CHK) Check internal consistency of directory. If an error occurs, the symbol table should be rebuilt. If B2(DD%RST) is also set, it is ignored; and the DELDF call must be executed again with B2(DD%RST) set to rebuild the symbol table.

AC2: Directory number

RETURNS +1: Always

The directory number given in AC2 must be that of the user's connected or logged-in directory unless the process has WHEEL or OPERATOR capability enabled, or the process has connect access to the directory being deleted.

If errors still occur after the symbol table is rebuilt, the process should restore the directory from magnetic tape; or the user should request help from the operator.

When a file with archive status is deleted and expunged, DELDF sends an IPCF message to GALAXY. This message contains all archive status information, which includes tape information, as well as the present file name, the user who expunged the file, and the time it was expunged.

Generates an illegal instruction interrupt on error conditions below.

DELDF ERROR MNEMONICS:

ARGX26: File is off line
DELDX1: WHEEL or OPERATOR capability required
DELDX2: Invalid directory number
DELFX2: File cannot be expunged because it is currently open
DELFX4: Directory symbol table could not be rebuilt
DELFX5: Directory symbol table needs rebuilding
DELFX6: Internal format of directory is incorrect
DELFX7: FDB formatted incorrectly; file not deleted
DELFX8: FDB not found; file not deleted
STRX10: Structure is offline

TOPS-20 MONITOR CALLS
(DELF)

Deletes the specified disk file and, if the file is closed, releases the JFN. The file is not expunged immediately, but is marked for later expunging either by the system or with the DELDF or LGOUT monitor calls.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: B0(DF%NRJ) Do not release the JFN.

B1(DF%EXP) Expunge the contents of the file. This also deletes the FDB entry in the directory. B0(DF%NRJ) and B1(DF%EXP) cannot be set simultaneously.

B2(DF%FGT) Expunge the file but do not deassign its addresses. The process must have WHEEL or OPERATOR capability enabled to set this bit. This bit should be set only by an operator or system specialist to delete a file that has a damaged or inconsistent index block.

B3(DF%DIR) Delete and expunge a directory file. The process must have WHEEL or OPERATOR capability enabled to set this bit. This bit should be set only by an operator or specialist to delete a bad directory.

B4(DF%ARC) Allow a file with archive status to be deleted.

B5(DF%CNO) Delete and expunge the contents of the file but preserve the file's name and FDB as they were (with the exception of the page count and the page table address). Setting this bit causes the DELF to fail if bit AR%NDL is set in word .FBBBT of the FDB, or if a complete set of tape back-up information is not in the FDB.

B18-35 JFN of the file being deleted.
(DF%JFN)

RETURNS +1: Failure, error code in AC1

+2: Success, JFN is released unless B0(DF%NRJ) is on or the file is open.

TOPS-20 MONITOR CALLS
(DELDF)

By setting B0(DF%NRJ), the user can delete multiple files by giving a JFN to GNJFN that represents a group of files and processing each file in the group.

The DELDF call takes the +1 return if the JFN is assigned to a nondirectory device.

DELDF ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DESX9: Invalid operation for this device
DELDFX1: Delete access required
DELDFX2: File cannot be expunged because it is currently opened
DELDFX3: System scratch area depleted; file not deleted
DELDFX4: Directory symbol table could not be rebuilt
DELDFX5: Directory symbol table needs rebuilding
DELDFX6: Internal format of directory is incorrect
DELDFX7: FDB formatted incorrectly; file not deleted
DELDFX8: FDB not found; file not deleted
DELDFX9: File is not a directory file
DELDF10: Directory still contains subdirectory
DLFX10: Cannot delete directory; file still mapped
DLFX11: Cannot delete directory file in this manner
DELX12: File has no pointer to offline storage
DELX13: File is marked "Never Delete"
STRX10: Structure is offline
WHELX1: WHEEL or OPERATOR capability required

Deletes all but the specified number of generations of a disk file. The files are marked for deletion and are expunged at a later time, either automatically by the system or explicitly with the DELDF or LGOUT call.

ACCEPTS IN AC1: B0(DF%NRJ) Do not release the JFN
B4(DF%ARC) Allow a file with archive status to be deleted.
B5(DF%CNO) Delete and expunge the contents of the file but preserve the file's name and FDB as they were (with the exception of the page count and the page table address). Setting this bit causes the DELDF to fail if bit AR%NDL is set in word .FBBT of the

TOPS-20 MONITOR CALLS
(DELNF)

FDB or if a complete set of tape backup information is not in the FDB.

B18-35 JFN of the file being deleted
(DF%JFN)

AC2: The number of generations to retain

RETURNS +1: Failure, error code in AC1

+2: Success, with the number of files deleted in AC2

Starting at the file specified by the JFN, the DELNF call decrements the generation number, first retaining the specified number of generations before deleting the remaining generations.

DELNF ERROR MNEMONICS:

DELX13: File is marked "Never Delete"
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DELFX1: Delete access required
STRX10: Structure is offline

Removes a request for a specific resource from the queue associated with that resource. The request is removed whether the process has a lock for the resource, or is only waiting in the queue for the resource.

This call can be used to remove any number of requests. If one of the requests cannot be dequeued, the dequeuing procedure continues until all requests that can be dequeued have been. An error return is given for the last request found that could not be dequeued. The process can then execute the ENQC call to determine the current status of each request. However, if the process attempts to dequeue more pooled resources than it originally allocated, the error return is taken and none of the pooled resources are dequeued.

See the TOPS-20 Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

RESTRICTIONS: Some functions require enabled WHEEL or OPERATOR capability to release system resource locks, or enabled WHEEL, OPERATOR, or ENQ capability to release global resource locks.

TOPS-20 MONITOR CALLS
(DEQ)

When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Function code

AC2: Address of argument block (required only for the .DEQDR function)

RETURNS +1: Failure, error code in AC1

+2: Success

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .DEQDR | Remove the specified requests from the queue. This function is the only one requiring an argument block. |
| 1 | .DEQDA | Remove all requests for this process from the queues. This action is taken on a RESET or LGOUT call. The error return is taken if the process has not given an ENQ call. |
| 2 | .DEQID | Remove all requests that correspond to the specified request identifier(ID). This function allows the process to release a class of locks in one call without itemizing each lock in an argument block. It is useful when dequeuing in one call the same locks that were enqueued in one call. To use this function, the process places the 18-bit request ID in AC2. |

The format of the argument block for function .DEQDR is identical to that given on the ENQ call. (Refer to the ENQ monitor call description.) However, the .ENQID word of the argument block is not used on a DEQ call and must be zero.

DEQ ERROR MNEMONICS:

DESX5: File is not open

ENQX1: Invalid function

ENQX2: Level number too small

ENQX3: Request and lock level numbers do not match

ENQX4: Number of pool and lock resources do not match

ENQX6: Requested locks are not all locked

ENQX7: No ENQ on this lock

ENQX9: Invalid number of blocks specified

ENQX10: Invalid argument block length

TOPS-20 MONITOR CALLS
(DEQ)

ENQX11: Invalid software interrupt channel number
ENQX13: Indirect or indexed byte pointer not allowed
ENQX14: Invalid byte size
ENQX15: ENQ/DEQ capability required
ENQX16: WHEEL or OPERATOR capability required
ENQX17: Invalid JFN
ENQX18: Quota exceeded
ENQX19: String too long
ENQX20: Locked JFN cannot be closed
ENQX21: Job is not logged in
DESX8: File is not on disk

Translates the given device designator to its corresponding ASCII device name string. The string returned contains only the alphanumeric device name; it does not contain a colon.

ACCEPTS IN AC1: Destination designator

AC2: Device designator

RETURNS +1: Failure, error code in AC1

+2: Success, updated string pointer in AC1, if pertinent

The STDEV monitor call can be used to translate a string to its corresponding device designator.

DEVST ERROR MNEMONICS:

DEVX1: Invalid device designator
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Inputs a double-precision, floating-point number, rounding if necessary.

ACCEPTS IN AC1: Source designator

RETURNS +1: Failure, error code in AC4 and updated string pointer in AC1, if pertinent.

TOPS-20 MONITOR CALLS
(DFIN)

+2: Success, double-precision, floating-point number in AC2 and AC3 and updated string pointer in AC1, if pertinent.

DFIN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
FLINX1: First character is not blank or numeric
FLINX2: Number too small
FLINX3: Number too large
FLINX4: Invalid format

Outputs a double-precision, floating-point number.

ACCEPTS IN AC1: Destination designator

AC2: First word of a normalized, double-precision, floating-point number

AC3: Second word of a normalized, double-precision, floating-point number

AC4: Format control word. (See Section 2.9.1.2.)

RETURNS +1: Failure, error code in AC4 and updated string pointer in AC1, if pertinent.

+2: Success, updated string pointer in AC1, if pertinent.

DFOUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
FLOTX1: Column overflow in field 1 or 2
FLOTX2: Column overflow in field 3
FLOTX3: Invalid format specified
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

TOPS-20 MONITOR CALLS
(DIAG)

WARNING: This JSYS can cause a system crash. Use with extreme caution.

NOTE

This JSYS is primarily intended for system use. The information returned may change in a future release.

Reserves a channel and either a single device or all devices attached to that channel. This call is also used to release the channel and its devices. When the request is made, no new activity is initiated on the requested channel, and the monitor waits for current activity on all devices connected to the channel to be completed. When the channel becomes idle, the process requesting the channel continues running.

The DIAG JSYS can also be used to get and release memory. The .DGGEM function is used by the system program TGHA for performing its spare bit substitution.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Negative length of the argument block in the left half, and address of the argument block in the right half.

RETURNS +1: Failure, error code in AC1
+2: Success

The available functions are as follows:

| Function | Symbol | Meaning |
|----------|--------|--|
| 1 | .DGACU | Assign the channel and a single device. Release the device after the time limit specified. |
| | | Word Contents |
| | | 0 function code |
| | | 1 device address |
| | | 2 time limit in milliseconds |
| 2 | .DGACH | Assign the channel and all devices. |
| | | Word Contents |
| | | 0 function code |
| | | 1 device address |

TOPS-20 MONITOR CALLS
(DIAG)

| | | |
|------|--------|--|
| 3 | .DGRCH | Release the channel and all assigned devices. |
| | Word | Contents |
| | 0 | function code |
| | 1 | device address |
| 4 | .DGSCP | Set up the channel program. The data transfer can be up to 50 pages. This function locks in memory the user page to which the channel control word points. This function also causes the system to update the Exec Process Table location corresponding to the channel with the appropriate channel control word (physical address). |
| | Word | Contents |
| | 0 | function code |
| | 1 | device address |
| | 2 | channel control word 0 |
| | 3 | channel control word 1 |
| | | . |
| | | . |
| | | . |
| | n+2 | channel control word n |
| 5 | .DGRCP | Release the channel program. The page for the specified channel, to which page the channel control word points, is unlocked. This function is not required before specifying a new channel program. |
| | Word | Contents |
| | 0 | function code |
| | 1 | device address |
| 6 | .DGGCS | Return the status of the channel. The argument block contains the logout area for the channel. |
| | Word | Contents |
| | 0 | function code |
| | 1 | device address |
| | 2-5 | 4-word channel logout area |
| 7-77 | | Reserved for DIGITAL. |
| 100 | .DGGEM | Get memory (for TGHA). |

TOPS-20 MONITOR CALLS
(DIAG)

| Word | Contents |
|------|---|
| 0 | function code |
| 1 | first page in user address space |
| 2 | first physical memory page |
| 3 | number of pages |
| 4 | user address of AR/ARX parity trap routines |

Upon successful return, this function accomplishes the following:

1. TOPS-20 has requested that all of the front ends refrain from accessing common memory.
2. The hardware PI system has been turned off; no scheduling can occur.
3. The time base and interval timer have been turned off.
4. All DTE byte transfers have been completed.
5. All RH20 activity has ceased.
6. The designated pages of the process address space have been set up to address the designated physical memory. Note that this is not the same as requesting the pages with PLOCK. With the get memory function, the data in the physical memory pages have been retained, and ownership of the pages is unchanged.
7. The CST0 entries for each of the designated physical pages have been saved and set as follows:
 - a) The age is set to the present age of the requesting process.
 - b) The process use field is set to all ones.
 - c) The modified bit is set to one.
8. The entire address space of the requesting process has been locked in memory. (Actually, only the pages that existed at the time of the DIAG call are locked. Therefore, the process must ensure that all of the pages it needs exist and are private when DIAG is executed.)

TOPS-20 MONITOR CALLS
(DIAG)

9. The monitor has set up proper dispatch if TGHA specified an AR/ARX trap address.

101 .DGREM Release memory (for TGHA)

Word Contents

0 function code

102 .DGPDL Inform the monitor that a device previously unknown to it is now available for use (is now online). This function is used with devices interfaced through the DX20 (TX01, TX03, TX05, TU70, or TU72).

Argument block:

Word Contents

0 function code
1 primary channel number
2 primary unit number
3 primary controller number (-1 if no controller)
4 alternate channel number
5 alternate unit number (should be same as primary unit number)
6 alternate controller number (-1 if no controller)

103 .DGCSL Reserved for DIGITAL.

104 .DGUCD CI-20 microcode management.

Word Contents

0 function code
1 subfunction code

| Code | Symbol | Meaning |
|------|--------|------------------------------|
| 0 | .DGRIP | microcode reload in progress |
| 1 | .DGRLC | microcode reload complete |
| 2 | .DGDIP | microcode dump in progress |
| 3 | .DGDMC | microcode dump complete |

105 .DGRST Reset any remote system on the CI

TOPS-20 MONITOR CALLS
(DIAG)

| | |
|------|---|
| Word | Contents |
| 0 | function code |
| 1 | system address: channel,,node where channel (which CI) is 7 for a KL, and node is the CI node address |
| 2 | 0 to set the force-bit to 0; one to set the force-bit to 1. Normally, a remote system will only allow itself to be reset by the system on the CI that did a previous reset of this system. The force-bit allows the calling system to force a reset whether or not it did the previous reset of the remote system. |

Note: Remote system may not support this function.

106 .DGSTR Start remote system

| | |
|------|---|
| Word | Contents |
| 0 | function code |
| 1 | system address: channel,,node where channel (which CI) is 7 for a KL, and node is the CI node address |
| 2 | 0 to use default start address of remote system; or start address for remote system if other than default |

Note: Remote system may not support this function.

107 .DGCTR Port counter functions

| | |
|------|---|
| Word | Contents |
| 0 | function code |
| 1 | channel,,function For the CI-20 (KLIPA), the channel is 7. |

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .DGGTC | get counters |
| 1 | .DGGVC | release counters |
| 2 | .DGPTC | set counters. This function will set the nodes to capture data and the data to capture. Note: .DGCTR function 0 (.DGGTC) must be executed prior to .DGPTC. |
| 3 | .DGRDC | read counters |

TOPS-20 MONITOR CALLS
(DIAG)

2 If releasing counters, then

 0 = do not force release. Ownership of
 counters will be released only if
 current owner is current process.
 1 = force release ownership of counters.

 If setting counters, then mask,, threshold

3 nodes to capture data if setting counters.

Words 2 - 15 are returned only if port counter
function = 3.

2 counter,, process number of owner.
 Counter is incremented whenever the port
 counters are set (initial value =-1)

3 CI-20 microcode version

4 path 0 ACKs

5 path 0 NAKs

6 path 0 no responses

7 path 1 ACKs

8 path 1 NAKs

9 path 1 no responses

10 number of datagrams discarded

11 total number of transmits

12 total number of receives

13 node on which data is being collected

14 packets received with CRC errors

15 mover parity errors,, CBUS parity errors

16 register PLIPE errors,, DATA PLIPE errors

17 channel errors,, EBUS parity errors

18 spurious channel errors,, CBUS available
 timeouts

19 spurious receive attentions,, spurious
 transmit attentions

20 transmit buffer parity errors,, transmit
 timeouts

110 .DGRSC Read SPEAR counter (the number of SPEAR packets
 queued to be written to the error file). The
 calling program should execute this function both
 before and after running any diagnostic test. If
 the value of the SPEAR counter changes, then SPEAR
 entries have been produced, some of which may be
 relevant to the diagnostic. This counter is never
 reset and never decremented.

Word Contents

0 function code

1 returned value of SPEAR counter

TOPS-20 MONITOR CALLS
(DIAG)

111 .DGENB Enable/disable use of .DGACH (assign controller and all devices). This function allows a diagnostic to gain control of the CI by allowing it to assign the CI to itself for the duration of the test. When the diagnostic has completed its testing, it should issue DIAG% function .DGRCH (release channel) and then issue .DGENB a second time to make the CI available to the system.

| Word | Contents |
|------|---|
| 0 | function code |
| 1 | RH20 slot number (7 for CI-20) |
| 2 | 0 to disable .DGACH and prevent further interruption of CI availability to system; -1 to enable .DGACH |

112 .DGWMD Write maintenance data to a remote node

| Word | Contents |
|------|---|
| 0 | function code |
| 1 | channel number |
| 2 | number of 8-bit bytes to be written |
| 3 | address in remote node to write data to |
| 4 | address of date to be written |

Note: Remote system may not support this function.

113 .DGRMD Read maintenance data from a remote node

| Word | Contents |
|------|--|
| 0 | function code |
| 1 | channel number |
| 2 | number of 8-bit bytes to be read |
| 3 | address in remote node to read data from |
| 4 | address to which data should be written |

Note: Remote system may not support this function.

The device address given in some of the argument blocks is a machine-dependent specification for the channel and device to be assigned. The devices that can be assigned must be attached to the RH20 controller and must be mounted by a process with either WHEEL, OPERATOR, or MAINTENANCE capability enabled. The format of the device address word is:

TOPS-20 MONITOR CALLS
(DIAG)

```

0          2 3          9 10          23 24          29 30          35
!=====!
! address ! device !    0    ! unit  ! subunit !
!  type   ! code   !      !      !        !
!=====!

```

DIAG ERROR MNEMONICS:

```

DIAGX1:  Invalid function
DIAGX2:  Device is not assigned
DIAGX3:  Argument block too small
DIAGX4:  Invalid device type
DIAGX5:  WHEEL, OPERATOR, or MAINTENANCE capability required
DIAGX6:  Invalid channel command list
DIAGX7:  Illegal to do I/O across page boundary
DIAGX8:  No such device
DIAGX9:  Unit does not exist
DIAG10:  Subunit does not exist
DIAG11:  Device is already on-line

```

Dismisses the process until the designated file input buffer is empty.

ACCEPTS IN AC1: File designator

RETURNS +1: Always

Returns immediately if the designator is not associated with a terminal.

The DOBE monitor call can be used to dismiss the process until the designated file output buffer is empty.

Generates an illegal instruction interrupt on error conditions below.

DIBE ERROR MNEMONICS:

```

DESX1:  Invalid source/destination designator
DESX3:  JFN is not assigned
DESX5:  File is not open
DEVX2:  Device already assigned to another job
TTYX01: Line is not active

```

TOPS-20 MONITOR CALLS
(DIC)

Deactivates the specified software interrupt channels. (See Section 2.6.1.)

ACCEPTS IN AC1: Process handle

AC2: 36-bit word
Bit n means deactivate channel n

RETURNS +1: Always

Software interrupt requests to deactivated channels are ignored except for interrupts generated on panic channels. Panic channel interrupts are passed to the closest superior process that has the specific channel enabled.

The AIC monitor call is used to activate specified software interrupt channels.

Generates an illegal instruction interrupt on error conditions below.

DIC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Disables the software interrupt system for a process.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always

If software interrupt requests are generated while the interrupt system is disabled, the requests are remembered and take effect when the interrupt system is reenabled unless an intervening CIS call is executed. However, interrupts on panic channels will still be generated even though the system is disabled.

In addition, if the CTRL/C terminal code is assigned to a channel, it will still generate an interrupt that cannot be disabled with a DIR call. CTRL/C interrupts can be disabled by deactivating the channel to which the code is assigned or by monitor action.

The EIR monitor call can be used to enable the software interrupt system for a process.

TOPS-20 MONITOR CALLS
(DIR)

Generates an illegal instruction interrupt on error conditions below.

DIR ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Translates the specified 36-bit user or directory number to its corresponding string and writes it to the given destination. When a user number is given, the string returned is the corresponding user name without any punctuation. When a directory number is given, the string returned is the corresponding structure and directory name including punctuation (structure:<directory>).

ACCEPTS IN AC1: Destination designator

AC2: User or directory number

RETURNS +1: Failure, with error code in AC1.

+2: Success, string written to destination, updated string pointer, if pertinent, in AC1

The RCDIR monitor call can be used to translate a directory string to its corresponding directory number. The RCUSR monitor call can be used to translate a user name string to its corresponding user number.

DIRST ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
DELFX6: Internal format of directory is incorrect
DIRX1: Invalid directory number
DIRX2: Insufficient system resources
DIRX3: Internal format of directory is incorrect
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged
STRX01: Structure is not mounted
STRX10: Structure is offline

TOPS-20 MONITOR CALLS
(DISMS)

Dismisses this process for the specified amount of time.

ACCEPTS IN AC1: Number of milliseconds for which the process is to be dismissed

RETURNS +1: When the elapsed time is up

The maximum argument specifiable in AC1 is 400,,0 (18 hours, 38 minutes, 28 seconds, and 864 milliseconds). If this value is exceeded, the argument is ignored and the maximum dismiss time is used. The time resolution is limited to the scheduling frequency (about 20 milliseconds).

Manipulates the Dump-on-BUGCHK facility which provides information on non-fatal system errors.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENENCE privileges.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Success

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .DBCNT | RH - Count of words in argument block, including this word. |
| 1 | .DBFNC | Function code. |
| 2-n | | Function specific arguments. |

The function codes for .DBFNC and their arguments are:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .DBENA | Enable DOB |
| 1 | .DBDIS | Disable DOB |
| 2 | .DBSBG | Set configuration word for a particular BUGxxx. |

Possible words and their configurations are:

TOPS-20 MONITOR CALLS
(DOB%)

| | | | |
|---|--------|-------------|---|
| | | Word | Contents |
| | | 2 (.DBNAM) | Name of the BUG in SIXBIT. |
| | | 3 (.DBCFIG) | New configuration word, defined as follows: |
| | | | B0(DB%ENA) - if on, set the bits to 1. If off, set the bits to 0. |
| | | | B1(DB%REQ) - request a dump on this BUG. |
| | | | B2(DB%IGN) - ignore timeout period for this BUG. |
| | | | B3(DB%DON) - (set by monitor) - BUG is dumped. |
| | | | B9(DB%NND) - (set by monitor) - BUG is not dumpable. |
| 3 | .DBPAR | | Enable/Disable DOB parameters |
| | | Word | Contents |
| | | 2 (.DBFLG) | B4(DB%INF) - Dump on all BUGINFs B5(DB%CHK) - Dump on all BUGCHKs |
| 4 | .DBIMD | | Take a dump immediately (FORCED BUGINF) |
| | | Word | Contents |
| | | 2 (.DBSTR) | Pointer to optional 7-bit string with structure name |
| 5 | .DBSTA | | Return the status of DOB. The status is returned starting in word .DBSTS of the argument block. (The minimum size of the block for this function is 2 words.) |
| | | Word | Contents |
| | | 2 (.DBSTS) | Appropriate flags: |
| | | | B0(DB%DOB) - DOB is enabled |
| | | | B4(DB%CHK) - dumps are requested for all BUGCHKs |
| | | | B5(DB%INF) - dumps are requested for all BUGINFs |
| | | | B6(DB%DIP) - dump is in progress |
| | | | B7(DB%ERR) - dump in progress had an I/O error |
| | | | B8(DB%DML) - DUMP.EXE file chosen for |

TOPS-20 MONITOR CALLS
(DOB%)

this dump was too small
for memory size of this
system.

3 (.DBNUM) Number of bugs for which dumping is
requested,,Number of bugs returned

4 (.DBTOV) Timeout value in seconds

The following two words are repeated
for each BUG returned:

5 (.DBBNM) SIXBIT BUG name

6 (.DBBCF) BUG configuration word

If the size of the user's block is 3, DOB% only
returns words 2 and 3 to the user (the status word
and the number of bugs requested). This enables a
user to determine how big an argument block is
needed for the call.

6 .DBTIM Set timeout value. Prevents continuous dumps from
occurring within the timeout period. By default,
this timer is set to 15 seconds.

Word Contents

2 (.DBTVS) Timeout value in seconds

Generates an illegal instruction interrupt on error conditions below.

DOB% ERROR MNEMONICS:

ARGX02: Invalid function
ARGX03: Illegal to change specified bits
ARGX04: Argument block too small
ARGX17: Invalid argument block length
CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required
DOBX01: Not a BUGCHK or BUGINF
DOBX02: DOB is disabled
DOBX03: DOB already disabled
DOBX04: DOB already enabled
DOBX05: Dump was not requested for this BUG
DOBX06: Dump was already requested for this BUG
DOBX07: Structure is not dumpable
DOBX08: DOB timeout out of range
STRX01: Structure is not mounted
STRX10: Structure is offline

TOPS-20 MONITOR CALLS
(DOBE)

Dismisses the process until the designated file output buffer is empty.

ACCEPTS IN AC1: Destination designator

RETURNS +1: Always

Returns immediately if designator is not associated with a terminal.

The DIBE monitor call can be used to dismiss the process until the designated file input buffer is empty.

Generates an illegal instruction interrupt on error conditions below.

DOBE ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Assigns or deassigns specific disk addresses.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: B0(DA%DEA) Deassign the specified address. If the address is currently assigned, control returns to the next instruction following the call (+1 return). If the address was not previously assigned, a BUGCHK occurs.

B1(DA%ASF) Assign a free page near the specified address. Assignment is on the same cylinder as the specified address, if possible, or on a nearby cylinder. If the specified address is 0, a page is assigned on a cylinder that is at least one-half free. If the assignment is not possible because the disk is full, control returns to the next instruction following the call.

B2(DA%CNV) Convert the specified address according to the setting of B3(DA%HWA).

TOPS-20 MONITOR CALLS
(DSKAS)

B3(DA%HWA) The specified address is a hardware address. If this bit is off, the specified address is a software address.

B4(DA%INI) Initialize a private copy of the bit table.

B5(DA%WRT) Write the private copy of the bit table to a new bit table file.

B6(DA%AIN) Abort the initialization of a private copy of the bit table.

B18-35 Disk address
(DA%ADR)

AC2: Device designator of structure. If DA%CNV is on in AC1, this argument is not required.

RETURNS +1: Failure, address already assigned or cannot be assigned

 +2: Success, address assigned in AC1

Generates an illegal instruction interrupt on error conditions below.

DSKAS ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

Allows the process to reference physical disk addresses when performing disk transfers. This monitor call requires the process to have WHEEL, OPERATOR, or MAINTENANCE capability enabled to read and write data. However, a process with only MAINTENANCE capability enabled can write data only if it is using physical addresses (.DOPPU) and writing to a unit that is not part of a mounted structure.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled. Some functions can be performed with MAINTENANCE capability enabled.

ACCEPTS IN AC1: B0-1(DOP%AT) Field indicating the address type. For physical channel and unit addresses, the value of the field is 1(.DOPPU) and the remainder of AC1 is:

 B2-6(DOP%CN) channel number

TOPS-20 MONITOR CALLS
(DSKOP)

B7-12(DOP%UN) unit number
B13-35(DOP%UA) unit address

For physical channel, controller, and unit numbers, refer to AC4.

For a structure and a relative address, the value of the field is 2(.DOPSR) and the remainder of AC1 is:

B2-10(DOP%SN) structure designator flag (0 is public structure). A value of -1 means the structure is indicated by the structure designator (see Section 2.4) in AC4.

B11-35(DOP%RA) relative address

Any other values for this field are illegal.

AC2: Control flags in the left half and a count of the number of words to transfer in the right half. The control flags are:

B9(DOP%NF) use values in AC4 for channel, controller, and unit numbers; otherwise, use values in AC1 (note: this bit must be on if DUP%AT has value .DOPSR).

B10(DOP%EO) error if unit offline. (Note that this is always the case if doing multi-paged transfers.)

B11(DOP%IL) inhibit error logging

B12(DOP%IR) inhibit error recovery

B13(DOP%PS) physical sector reference. Intended to permit homeblocks to be read/written when MSTR% JSYS function .MSRSP is not equal to MSTR% JSYS function .MSTSP.

B14(DOP%WR) write data to the disk. If this bit is off, read data from the disk.

B18-35 (DOP%CT) word count. If this count is less than or equal to 1000, the data to be transferred cannot straddle a page boundary. Thus the caller's buffer should start at a page boundary and cannot be longer than one page.

If this count is more than 1000, the data to be transferred can straddle a

TOPS-20 MONITOR CALLS
(DSKOP)

page boundary, so the caller's buffer need not start on a page boundary, and the buffer can be larger than one page. Two restrictions apply, however. First, the buffer must be a multiple of the size of the sectors on the disk being read or written. (Obtain the sector size by using the .MSRUS function of the MSTR JSYS.) Second, no error processing is done (the JSYS executes as though the DOP%IL and DOP%IR bits were set). On an error, the pages must be read one at a time to determine which pages caused errors.

AC3: Address in caller's address space from which data is written or into which data is read.

AC4: Device designator of the structure. This word is used if the value given for DOP%SN is -1.

or

Physical channel, controller, and unit numbers if B9(DOP%NF) in AC2 is on. In this case,

B0-11(DOP%C2) channel number
B12-23(DOP%K2) controller number
B24-35(DOP%U2) unit number

RETURNS +1: Always, AC1 is nonzero if an error occurred, or zero if no error occurred.

No more than 50 pages can be transferred at a time. In addition, a transfer cannot cross a cylinder boundary.

If an error occurs and DOP%IL is on in the call, no error logging is performed. If DOP%IL is off, the standard system error logging is performed.

If an error occurs and DOP%IR is on in the call, no retries or ECC corrections, if applicable, are attempted. If DOP%IR is off, the standard system error recovery procedure is followed.

An error occurs if the format for channel, controller, and unit number is used with Release 4 or any previous monitor.

Generates an illegal instruction interrupt on error conditions below.

DSKOP ERROR MNEMONICS:

DKOP01: Illegal disk address
DKOP02: Transfer too large

TOPS-20 MONITOR CALLS
(DSKOP)

DKOP03: Invalid unit specified
DKOP04: Illegal address specified
DKOP05: Size not sector size
DKOP06: Data or device error
DKOP07: Device is offline
DSKOX1: Channel number too large
DSKOX2: Unit number too large
DSKOX3: Invalid structure number
DSKOX4: Invalid address type specified
DECRSV: DEC-reserved bits not zero
WHELX1: WHEEL or OPERATOR capability required
STRX10: Structure is offline

Detaches the controlling terminal from the current job. (The ATACH call with bit 1 (AT%NAT) of AC2 set can be used to detach a job other than the current job.) A console-detached entry is appended to the accounting data file.

RETURNS +1: Always

The DTACH call is ignored if the job is already detached.

The ATACH monitor call is used to attach the controlling terminal to a specified job.

Deassigns a terminal interrupt code.

ACCEPTS IN AC1: Terminal interrupt code; see Section 2.6.6

RETURNS +1: Always

The DTI call is a no-op if the specified terminal code was not assigned by the current process.

The ATI monitor call is used to assign a terminal code.

Generates an illegal instruction interrupt on error conditions below.

DTI ERROR MNEMONICS:

TERMX1: Invalid terminal code

TOPS-20 MONITOR CALLS
(DUMPI)

Reads data words into memory in unbuffered data mode. The file must be open for data mode 17. (See Section 2.4.7.5 for information about unbuffered magnetic tape I/O.)

ACCEPTS IN AC1: JFN

AC2: B0(DM%NWT) Do not wait for completion of requested operation

B18-35 Address of command list in memory
(DM%PTR)

RETURNS +1: Failure, error code in AC1, pointer to offending command in AC2

+2: Success, pointer in AC2 updated to last command

The use of B0(DM%NWT) allows data operations to be double-buffered with a resulting increase in speed. When this bit is on, DUMPI/DUMPO returns immediately after the request is queued. This allows the program to overlap computations with I/O transfers. If the second request is then made, the program is blocked until the first request is completed. Generally, for a sequence of overlapped DUMPI/DUMPO calls, return from the Nth call indicates that the Nth-1 request has completed and that the Nth request is now in progress. This bit is implemented only for magnetic tape.

The GDSTS call can be used after the transfer is completed to determine the number of bytes read.

If an error occurs on the Nth request, the failure return is given on the Nth+1 call, and the Nth+1 request is ignored. This means that the program will discover an error on a request only after making the next request. The next request is ignored to prevent improper operation and must be reissued after the error has been processed. The GDSTS call can be executed to determine the cause for the error.

COMMAND LIST FORMAT:

Three types of entries may occur in the command list.

1. IOWD n, loc - Causes n words to be transferred from the file to locations loc through loc+n-1 of the process address space. The next command is obtained from the location following the IOWD. For magnetic-tape files, 1 IOWD word reads 1 physical tape record. For labeled magnetic-tape files, the data format must be "U".

The IOWD pseudo-op generates XWD -n,loc-1.

TOPS-20 MONITOR CALLS
(DUMPI)

2. XWD 0, y - Causes the next command to be taken from location y. Referred to as a GOTO word.
3. 0 - Terminates the command list.

DUMPI ERROR MNEMONICS:

DUMPX1: Command list error
DUMPX2: JFN is not open in dump mode
DUMPX3: Address error (too big or crosses end of memory)
DUMPX4: Access error (cannot read or write data in memory)
DUMPX5: No-wait dump mode not supported for this device
DUMPX6: Dump mode not supported for this device
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
IOX1: File is not opened for reading
IOX4: End of file reached
IOX5: Device or data error

Writes data words from memory in unbuffered data mode. The file must be open for data mode 17. (See Section 2.4.7.5 for information about unbuffered magnetic tape I/O.)

ACCEPTS IN AC1: JFN

AC2: B0(DM%NWT) Do not wait for completion of requested operation

B18-35 Address of command list in memory
(DM%PTR)

RETURNS +1: Failure, error code in AC1, pointer to offending command in AC2

+2: Success, pointer in AC2 updated to last command

This call locks in memory the pages to be transferred. Any attempt to write to these pages while DUMPO has them locked results in an illegal memory reference.

The use of B0(DM%NWT) allows data operations to be double-buffered with a resulting increase in speed. When this bit is on, DUMPI/DUMPO returns immediately after the request is queued. This allows the program to overlap computations with I/O transfers. If the second

TOPS-20 MONITOR CALLS
(DUMPO)

request is then made, the program is blocked until the first request is completed. Generally, for a sequence of overlapped DUMPI/DUMPO calls, return from the Nth call indicates that the Nth-1 request has completed and that the Nth request is now in progress. This bit is implemented only for magnetic tape.

COMMAND LIST FORMAT:

Three types of entries may occur in the command list.

1. IOWD n, loc - Causes n words from loc through loc+n-1 to be transferred from the process address space to the file. The next command is obtained from the location following the IOWD. For mag-tape files, 1 IOWD word writes 1 physical tape record. For labeled mag-tape files, the data format must be "U".

NOTE

Dump mode output to a labeled tape can override the block-size limit specified in the GTJFN. If any write produces a block in excess of the specified block-size parameter, then the file can be read only in dump mode.

The IOWD pseudo-op generates XWD -n,loc-1.

2. XWD 0, y - Causes the next command to be taken from location y. Referred to as a GOTO word.
3. 0 - Terminates the command list.

The GDSTS call can be used after the transfer is completed to determine the number of bytes written.

DUMPO ERROR MNEMONICS:

DUMPX1: Command list error
DUMPX2: JFN is not open in dump mode
DUMPX3: Address error (too big or crosses end of memory)
DUMPX4: Access error (cannot read or write data in memory)
DUMPX5: No-wait dump mode not supported for this device
DUMPX6: Dump mode not supported for this device
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
IOX2: File is not opened for writing
IOX5: Device or data error
IOX11: Quota exceeded

TOPS-20 MONITOR CALLS
(DUMPO)

IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Returns the characteristics of the specified device.

ACCEPTS IN AC1: JFN or device designator

RETURNS +1: Always, with

AC1: Containing the device designator (even if a JFN was given).
AC2: Containing the device characteristics word.
AC3: Containing the job number to which the device is assigned in the left half and the unit number in the right half. If the device is a structure or does not have units, the right half is -1.

The left half of AC3 contains -1 if the device is not assigned to any job or -2 if the device allocator has ownership of the device.

Device Characteristics Word

| Bit | Symbol | Meaning |
|------|--------|---|
| 0 | DV%OUT | device can do output |
| 1 | DV%IN | device can do input |
| 2 | DV%DIR | device has a directory |
| 3 | DV%AS | device is assignable with ASND |
| 4 | DV%MDD | device has multiple directories |
| 5 | DV%AV | device is available or assigned to this job |
| 6 | DV%ASN | device is assigned by ASND |
| 8 | DV%MNT | device is mounted |
| 9-17 | DV%TYP | device type |
| | | 0 .DVDSK disk |
| | | 2 .DVMTA magnetic tape |
| | | 7 .DVLPT line printer |
| | | 10 .DVCDR card reader |
| | | 11 .DVFE front-end pseudo-device |
| | | 12 .DVTTY terminal |
| | | 13 .DVPTY pseudo-terminal |
| | | 15 .DVNUL null device |
| | | 16 .DVNET ARPA network |
| | | 22 .DVDCN DECnet active component |
| | | 23 .DVSRV DECnet passive component |

TOPS-20 MONITOR CALLS
(DVCHR)

| | | | |
|-------|--------|---|-------------------|
| 18 | DV%PSD | device is a pseudo-device | |
| 20-35 | DV%MOD | data mode in which device can be opened | |
| | B20 | DV%M17 | dump mode |
| | B27 | DV%M10 | image mode |
| | B34 | DV%M1 | small buffer mode |
| | B35 | DV%M0 | normal mode |

Generates an illegal instruction interrupt on error conditions below.

DVCHR ERROR MNEMONICS:

DEVX1: Invalid device designator
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer

Enables the software interrupt system for a process. (See Section 2.4.)

ACCEPTS IN AC1: Process handle

RETURNS +1: Always

The DIR monitor call can be used to disable the software interrupt system for a process.

Generates an illegal instruction interrupt on error conditions below.

EIR ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Requests access to a specific resource by placing a request in the queue for that resource. This call can be used to request any number of resources.

Refer to the Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

TOPS-20 MONITOR CALLS
(ENQ)

RESTRICTIONS: Some functions require enabled WHEEL or OPERATOR capability to acquire system resource locks, or enabled WHEEL, OPERATOR, or ENQ capability to acquire global resource locks.

When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Function code

AC2: Address of argument block

RETURNS +1: Failure, error code in AC1

+2: Success

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .ENQBL | Queue the requests and block the process until all requested locks are acquired. The error return is taken only if the call is not correctly specified. |
| 1 | .ENQAA | Queue the requests and acquire the locks only if all requested resources are immediately available. No requests are queued and the error return is taken if any one of the resources is not available. |
| 2 | .ENQSI | Queue the requests. If all requested resources are immediately available, this function is identical to the .ENQBL function. If all resources are not immediately available, the request is queued and the call fails with the ENQX6 error. A software interrupt will occur when all requested resources have been given to the process. |
| 3 | .ENQMA | Modify the access of a previously queued request. (Refer to (Refer to EN%SHR below.) The access of each lock in this request is compared with the access of each lock in the previously queued request. If the two accesses are the same, no modification is needed or made. |

If the access in this request is shared and the access in the previous request is exclusive, the call succeeds. If the access in this request is exclusive and the access in the previous request

TOPS-20 MONITOR CALLS
(ENQ)

is shared, this function returns an error unless this process is the only user of the lock. If the caller is the only user of this lock, the call succeeds. The error return is also taken if:

1. Any one of the specified locks does not have a pending request.
2. Any one of the specified locks is a pooled resource.

This function checks each lock specified, and the access is changed for all locks that were given correctly. If the call fails, the user must execute the ENQC call to determine the current state of each lock.

- 4 .ENECL Enable cluster-wide ENQ/DEQ functionality for all subsequent ENQ%/DEQ%/ENQC% JSYSes done by this process. This function does not require an argument block and so the contents of AC2 are ignored.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .ENQLN | length of the header and the number of requested locks in the left half, and length of argument block in the right half. |
| 1 | .ENQID | the request ID in the left half, and the software interrupt channel number in the right half. |
| 2 | .ENQLV | flags and level number in the left half, and JFN, -1, -2, or -3 in the right half. (see word .ENQMS below) |
| 3 | .ENQUC | pointer to a string or a 5B2+33-bit user code. (see word .ENQMS below) |
| 4 | .ENQRS | number of resources in pool in the left half and number of resources requested in the right half, or 0 in the left half and a group number in the right half. (see word .ENQMS below) |
| 5 | .ENQMS | address of a resource mask block. |

Words .ENQLV through .ENQMS should be repeated for each resource requested.

TOPS-20 MONITOR CALLS
(ENQ)

The argument block is divided into two logical sections: a header and individual requests for each desired lock. Words .ENQLN and .ENQID form the header. Word .ENQLV through word .ENQMS form the individual request and are repeated for each lock being requested. The words in the argument block are described in the following paragraphs.

.ENQLN

The length of the header (.ENHLN) is contained in bits 0 through 5. Currently, the length of the header is two words. (Note that a given length of zero or one is assumed to be equal to a length of two.) The number of locks being requested (.ENNLK) is contained in bits 6 through 17, and the length of the argument block (.ENALN) is contained in bits 18 through 35.

.ENQID

The software interrupt channel specifies the number of the channel on which to generate an interrupt with the .ENQSI function. The request ID is an 18-bit user-generated value used to identify the particular resource. This ID is not currently used by the system but, instead, is stored for future expansion of the facility.

.ENQLV

The following flags are defined:

- | | |
|------------|---|
| B0(EN%SHR) | Access to this resource is to be shared. If this bit is not set, access to the resource is to be exclusive. |
| B1(EN%BLN) | Ignore the level number associated with this resource. Sequencing errors in level numbers will not be considered fatal, and execution of the call will continue. If a sequencing error occurs, the successful return is taken, and AC1 will contain an error code indicating the sequencing error that occurred. |
| B2(EN%NST) | Allow ownership of this lock to be nested to any level within a process. This means that a process can request this resource again even though it already owns it. If the process has a request in the resource's queue or if the process already owns the lock, the ownership of the lock is nested to a depth one greater than the current depth. If the process does not have a request in the resource's queue, the setting of this bit has no effect, and the execution of the ENQ call continues. When a process has a nested lock, it must DEQ the resource as many times as it ENQed it before the resource becomes available to other processes. |

TOPS-20 MONITOR CALLS
(ENQ)

B3(EN%LTL) Allow a long-term lock on this resource. This notifies the system that this resource will be locked and unlocked many times in a short period of time. Setting this bit permits a program to run faster if it is doing multiple locks and unlocks on the same resource because the argument block data is not deleted immediately from the ENQ/DEQ data base when a DEQ call is executed. Thus, the time required to re-create the data is reduced.

B9-17(EN%LVL) Level number associated with this resource.

The request is not queued and the error return is taken if EN%BLN is not set and

1. A resource with a level number less than or equal to the highest numbered resource requested so far is specified.
2. The level number of the current request does not match the level number supplied on previous requests for this resource.

The right half of .ENQLV specifies the type of access desired for the resource. If a JFN is given, the file associated with the JFN is subject to the standard access protection of the system. The file associated with the JFN in the right half of .ENQLV must be opened before the ENQ is performed or an error will be generated. If -1 is given, the resource can be accessed only by processes of the job. If -2 is given, the resource can be accessed by any job on the system. (The process must have ENQ capability enabled to specify -2.) If -3 is given, the resource can be accessed only by processes that have WHEEL or OPERATOR capability enabled.

.ENQUC

This word is either a byte pointer or a 33-bit user code, either of which serves to uniquely identify the resource to all users. This quantity is the second part of the resource name. (JFN, -1, -2, or -3 is the first part of the resource name.) The system makes no association between these identifiers and any physical resource.

The string identified by the byte pointer can contain bytes of any size (from 1 to 36 bits), and is terminated by a null byte. The byte size is specified by the byte pointer. The maximum length of the string (including the terminating null byte) is 50 words.

TOPS-20 MONITOR CALLS
(ENQ)

.ENQRS

This word is used to allocate multiple resources from a pool of identical resources. The left half contains the number of resources in the pool, and is a parameter agreed upon by all users. All requests for the same pooled resource must agree with the original count or the call fails. The number of resources requested from the pool must be greater than zero if a pool exists, and must be less than or equal to the number in the pool.

If the left half of this word is zero, the system assumes only one resource of the specific type exists. In this case, if the right half of this word is positive, it is interpreted as the number of the group of users who can simultaneously access the resource.

.ENQMS

Obtains a single lock representing many specific resources. For example, a lock can be obtained on a particular data base, and the specific resources requested can be individual records in that data base.

This word contains an address of a mask block, consisting of a count word and a group of mask words. The first word of the mask block contains a count (in the right half-word) of the number of words in the block, including the count word. The remaining words each contain 36 mask bits, where each bit represents a specific resource of the lock. The maximum length of the mask block is 16 words. All requests for the resources associated with the mask block must specify the same length for the block or an error return is taken. Also, when a mask block is specified, the ENQ call must request exclusive access to the resource and the left half of word .ENQRS of the lock request must be zero.

The set of resources comprising the lock is a parameter agreed upon by all users. A process can obtain exclusive access to all or some of the specific resources comprising the lock. When a process requires exclusive access to all the resources, it executes an ENQ call (for exclusive access) and does not specify a mask block. A successful return is given if there are no other processes that have issued an ENQ call for that lock. Otherwise, the process blocks until the requested resources are available.

When a process requires exclusive access to some of the specific resources comprising the lock, it sets up the mask block and sets the bits corresponding to the specific resources it wants to lock. The process then executes an ENQ call for exclusive access. On successful execution of the ENQ call, the process has an exclusive lock for the resources represented by the bits on in the mask. The process blocks if another process owns an exclusive lock on the resource and that process's ENQ call has not specified a mask block.

TOPS-20 MONITOR CALLS
(ENQ)

Once a mask block has been set up for a set of specific resources, subsequent requests for a different set of resources will be honored. The set of resources being requested is considered different if the bits on in one process's mask block are not on in another process's mask block. When a subsequent request is given for resources that are currently locked by a process, the process with the request blocked until the last of the currently locked resources is dequeued by the owner of the lock.

A process can dequeue all or part of the original ENQ call request. When a DEQ call is executed, the bits on in the mask block of the DEQ call are compared with the bits on in the original ENQ call. The resources not being dequeued remain locked and must be dequeued by a subsequent DEQ call. This action allows a process to lock a number of resources all at once, and then to release individual resources as it finishes with them. However, a process cannot execute subsequent ENQ calls to request additional resources from those requested in its original ENQ call.

ENQ ERROR MNEMONICS:

| | |
|---------|--|
| DESX5: | File is not open |
| ENQX1: | Invalid function |
| ENQX2: | Level number too small |
| ENQX3: | Request and lock level numbers do not match |
| ENQX4: | Number of pool and lock resources do not match |
| ENQX5: | Lock already requested |
| ENQX6: | Requested locks are not all locked |
| ENQX7: | No ENQ on this lock |
| ENQX8: | Invalid access change requested |
| ENQX9: | Invalid number of blocks specified |
| ENQX10: | Invalid argument block length |
| ENQX11: | Invalid software interrupt channel number |
| ENQX12: | Invalid number of resources requested |
| ENQX13: | Indirect or indexed byte pointer not allowed |
| ENQX14: | Invalid byte size |
| ENQX15: | ENQ/DEQ capability required |
| ENQX16: | WHEEL or OPERATOR capability required |
| ENQX17: | Invalid JFN |
| ENQX18: | Quota exceeded |
| ENQX19: | String too long |
| ENQX20: | Locked JFN cannot be closed |
| ENQX21: | Job is not logged in |
| ENQX22: | Invalid mask block length |
| ENQX23: | Mismatched mask block lengths |
| ENQX24: | Internal resources exhausted (No more SCA buffers) |
| DESX8: | File is not on disk |

TOPS-20 MONITOR CALLS
(ENQC)

Returns the current status of the given resource and obtains information about the state of the queues. This monitor call also allows privileged processes to manipulate access rights to the queues and to perform other utility functions on the queue structure.

Refer to the Monitor Calls User's Guide for an overview and description of the Enqueue/Dequeue facility.

The ENQC monitor call has two calling sequences, depending on whether the process is obtaining status information or is modifying the queue structure.

Obtaining Status Information

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Function code (.ENQCS)

AC2: Address of argument block

AC3: Address of block in which to place status

RETURNS +1: Failure, error code in AC1

+2: Success

The function .ENQCS returns the status of the specified resources.

The argument block is identical in format to the ENQ and DEQ argument blocks. (Refer to the ENQ monitor call description.)

The status block has a 3-word entry for each resource specified in the argument block. This entry reflects the current status of the resource and has the following format:

```

0                               17 18                               35
!=====!
!           flag bits indicating status of resource           !
!=====!
!                               36-bit time stamp                               !
!=====!
! # of processes with lock !           request ID           !
!=====!

```

TOPS-20 MONITOR CALLS
(ENQC)

The following flag bits are currently defined.

- B0(EN%QCE) An error has occurred in the corresponding resource request and bits 18-35 contain an appropriate error code.
- B1(EN%QCO) This process owns the lock.
B2(EN%QCQ) This process is in the queue waiting for this resource. This bit is set if B1(EN%QCO) is set because a request remains in the queue until a DEQ call is given.
- B3(EN%QCX) The lock has been allocated for exclusive access.
- B4(EN%QCB) This process is in the queue waiting for exclusive access to the resource. This bit is off if B2(EN%QCQ) is off.
- B9-17(EN%LVL) The level number of the resource.
- B18-35(EN%JOB) Global job number of the owner of the lock. This value may be a job number on another system within the cluster. For locks with shared access, this value will be the job number of one of the sharers. However, this value will be the current job's number if the current job is one of the sharers. If the lock is not owned, the value is -1. If B0(EN%QCE) is on, this field contains the appropriate error code.

The time stamp indicates the last time a process was given access to the resource. The time is in the universal date-time standard. If no process currently has access to the resource, the word is zero.

The number returned in the left half of the third word indicates the number of processes that currently have the resource locked for either exclusive access or shared access.

The request ID is either the request ID of the current process if that process is in the queue, or the request ID of the owner of the lock.

Modifying the Queue Structure

RESTRICTIONS: These functions require enabled WHEEL or OPERATOR capability.

When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Function code

TOPS-20 MONITOR CALLS
(ENQC)

AC2: Address of argument block

RETURNS +1: Failure, error code in AC1
+2: Success

The available functions, along with their argument block formats, are as follows:

| Function | Argument Block | Meaning |
|----------|--|---|
| .ENQCG | One word containing a job number in the right half. The left half is ignored. | Return the ENQ/DEQ quota for the specified job. The quota is returned in AC1. A job number of -1 defines your own job. |
| .ENQCC | the new quota in the left half and a job number in the right half. | One word containing Change the ENQ/DEQ quota for the specified job. The process executing the call must have WHEEL capability enabled or an error code is returned. |
| .ENQCD | The first word is the length of the block (n). Remaining words contain the returned data. (See below.) | A block of n words. Dump the ENQ/DEQ locks and queue entries into the argument block. The process executing the call must have WHEEL capability enabled or an error code is returned. |

The data returned in the argument block concerns both the ENQ/DEQ locks and the queues. The data concerning the locks is in a 4-word block of the following format:

| | 0 | 8 9 | 17 18 | 35 |
|--------|--|----------------------------|-------------------------------|----|
| .ENQDF | ! flags | ! level number | ! OFN, 40000+job#, -2, or -3! | ! |
| .ENQDR | ! total resources in pool | ! # of resources remaining | ! | ! |
| .ENQDT | ! time stamp of last request | ! locked | ! | ! |
| .ENQDC | ! user code of lock or beginning of string | ! | ! | ! |

TOPS-20 MONITOR CALLS
(ENQC)

If there are no pooled resources, word .ENQDR has the format:

```

          0                               17 18                               35
          !=====!
.ENQDR  !           0           !           group number           !
          !=====!

```

The data concerning the queues is in a 2-word block of the following format:

```

          0           8 9           17 18                               35
          !=====!
.ENQDF  !   flags   !software chan! job # creator queue entry !
          !=====!
.ENQDI  !group # or number requested!           request ID           !
          !=====!

```

The flags returned in the first word of each block are as follows:

- B0(EN%QCL) This block concerns data about the locks. If this bit is off, the block concerns data about the queues.
- B1(EN%QCO) This process owns the lock.
- B2(EN%QCT) This lock contains a text string.
- B3(EN%QCX) This lock is for exclusive access.
- B4(EN%QCB) This process is blocked until exclusive access is available.
- B5(EN%QCC) This is a cluster-wide lock/request.
- B6(EN%QCN) This lock requires no vote.
- B7(EN%QCS) This lock requires a scheduling pass.

ENQC ERROR MNEMONICS:

- ENQX1: Invalid function
- ENQX2: Level number too small
- ENQX3: Request and lock level numbers do not match
- ENQX4: Number of pool and lock resources do not match
- ENQX5: Lock already requested
- ENQX6: Requested locks are not all locked
- ENQX7: No ENQ on this lock
- ENQX8: Invalid access change requested
- ENQX9: Invalid number of blocks specified
- ENQX10: Invalid argument block length

TOPS-20 MONITOR CALLS
(ENQC)

ENQX11: Invalid software interrupt channel number
ENQX12: Invalid number of resources requested
ENQX13: Indirect or indexed byte pointer not allowed
ENQX14: Invalid byte size
ENQX15: ENQ/DEQ capability required
ENQX16: WHEEL or OPERATOR capability required
ENQX17: Invalid JFN
ENQX18: Quota exceeded
ENQX19: String too long
ENQX20: Locked JFN cannot be closed
ENQX21: Job is not logged in
ENQX24: Internal resources exhausted (No more SCA buffers)
DESX8: File is not on disk

Enables the capabilities for the specified process. (Refer to Section 2.7.1 for a description of the capability word.)

ACCEPTS IN AC1: Process handle
AC2: Capabilities the process can enable
AC3: Capabilities to enable

RETURNS +1: Always

The capabilities in bits 0-8 and bits 18-35 of AC2 are matched (ANDed) with the corresponding capabilities of both the calling process and the process specified in AC1. The calling process can only enable those capabilities that both the calling process and the object process have.

The contents of AC2 are ignored if the process handle in AC1 is for the current process.

The RPCAP monitor call can be used to obtain the capabilities of a process.

Generates an illegal instruction interrupt on the following error conditions:

EPCAP ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process

TOPS-20 MONITOR CALLS
(ERSTR)

Translates a TOPS-20 error number to its corresponding text string and writes the string to the specified destination. This error number is the one returned in an AC (usually in AC1) on a JSYS error and is associated with a unique error mnemonic and text string. The error numbers begin at 600010 and are defined in the system file MONSYM.MAC. (Refer to Appendix B for the list of error numbers, mnemonics, and text strings.)

ACCEPTS IN AC1: Destination designator

AC2: LH: Process handle
RH: Error number, or -1 for the most recent error in the specified process. If an error number is specified, .FHSLF should be specified in the left half of AC2.

AC3: LH: A negative count of the maximum number of bytes in the string to be transferred, or 0 for no limit
RH: 0

RETURNS +1: Failure, undefined error number
+2: Failure, string size out of bounds or invalid destination designator
+3: Success, with updated byte pointer in AC1

Generates an illegal instruction interrupt on error conditions below.

ERSTR ERROR MNEMONICS:

DESX1: Invalid source/destination designator
FRKHX1: Invalid process handle
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Outputs an error string. This monitor call reports an error in the primary input stream, and resynchronizes the input transaction. This mechanism is convenient for communicating with a user who made a typing error and may have continued to type. It also allows error messages to have a standard format.

ACCEPTS IN AC1: Byte pointer to a string in the caller's address

TOPS-20 MONITOR CALLS
(ESOUT)

space. The string is terminated with a null character.

RETURNS +1: Always, with updated byte pointer in AC1

The ESOUT call waits for the primary output buffer to empty and then outputs a carriage return, line feed, and question mark to the primary output designator. Next, it clears the primary input buffer and outputs the error string to the primary output device.

Can cause several software interrupts or process terminations on certain file conditions. (Refer to bit OF%HER of the OPENF call description.)

Finds the first free page in the specified file. A free page is one that is marked as not being in use. The FFFFP call is useful for finding a nonused page in a file before a PMAP call is executed that writes into that page.

ACCEPTS IN AC1: Starting page number in left half, JFN in right half.

RETURNS +1: Always, with the JFN in the left half of AC1 and the page number in the right half of AC1, or a fullword -1 in AC1 if there is no free page.

Generates an illegal instruction interrupt on the following error conditions:

FFFFP ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Illegal use of terminal designator or string pointer
DESX5: File is not open

Freezes one or more processes.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always

This suspends the processes (as soon as they are stoppable from the monitor's point of view) in such a way that they can be continued at

TOPS-20 MONITOR CALLS
(FFORK)

the place they were suspended. However, they do not have to be continued; they could be killed.

The FFORK call is ignored if the referenced process is already frozen.

The RFORK monitor call can be used to resume one or more processes.

Generates an illegal instruction interrupt on the following error conditions:

FFORK ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Finds the first used page of the file at or beyond the specified page number.

ACCEPTS IN AC1: JFN in the left half, and the starting page number in the right half

RETURNS +1: Failure, error code in AC1
+2: Success, page number in the right half of AC1. The left half of AC1 is unchanged.

FFUFP ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Illegal use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
FFUFX1: File is not open
FFUFX2: File is not on multiple-directory device
FFUFX3: No used page found

Inputs a floating-point number from the specified source. This call ignores leading spaces and terminates on the first character that cannot be part of a floating point number. If that character is a carriage return followed by a line feed, the line feed is also input.

ACCEPTS IN AC1: Source designator

TOPS-20 MONITOR CALLS
(FLIN)

RETURNS +1: Failure, error code in AC3 and updated string pointer
 in AC1, if pertinent

 +2: Success, single-precision, floating-point number in
 AC2 and updated string pointer in AC1, if pertinent

FLIN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: file is not open
FLINX1: first character is not blank or numeric
FLINX2: number too small
FLINX3: number too large
FLINX4: invalid format

Outputs a floating-point number to the specified destination.

ACCEPTS IN AC1: Destination designator

 AC2: Normalized, single-precision, floating-point number

 AC3: Format control word. (Refer to Section 2.9.1.2.)

RETURNS +1: Failure, error code in AC3 and updated string pointer
 in AC1, if pertinent

 +2: Success, updated string pointer in AC1, if pertinent

FLOUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: File is not open
FLOTX1: Column overflow in field 1 or 2
FLOTX2: Column overflow in field 3
FLOTX3: Invalid format specified
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

TOPS-20 MONITOR CALLS
(GACCT)

Returns the current account for the specified job.

RESTRICTIONS: Some functions require Confidential Information Access, WHEEL, or OPERATOR capability enabled.

ACCEPTS IN AC1: Job number, or -1 for current job

AC2: Byte pointer to string where alphanumeric account designator (if any) is to be stored

RETURNS +1: Always, with updated pointer to account string in AC2

The GACCT monitor call requires the process to have Confidential Information Access, WHEEL, or OPERATOR capability enabled if the specified job number is not for the current job.

The CACCT monitor call can be used to change the account for the current job.

Generates an illegal instruction interrupt on the following error conditions:

GACCT ERROR MNEMONICS:

GACCX1: Invalid job number

GACCX2: No such job

GACCX3: Confidential Information Access capability required

Returns the account designator to which the specified file is being charged.

ACCEPTS IN AC1: JFN

AC2: Byte pointer to string in caller's address space where account string (if any) is to be stored

RETURNS +1: Failure, error code in AC1

+2: Success, account string returned, updated string pointer in AC2

+3: Success, 5B2+account number returned in AC2

The SACTF monitor call can be used to set the account designator to which the file is to be charged.

TOPS-20 MONITOR CALLS
(GACTF)

GACTF ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
GACTX1: File is not on multiple-directory device
GACTX2: File expunged
GACTX3: Internal format of directory is incorrect

Returns the entry vector and the UWO locations for the compatibility package.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with entry vector length in the left half and entry vector address in the right half of AC2, and UWO location in the left half and PC location in the right half of AC3.

If use of the compatibility package has been disabled, AC2 contains -1 on return. If the compatibility package is not available, AC2 and AC3 contain 0 on return.

The SCVEC monitor call can be used to set the entry vector for the compatibility package.

GCVEC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

Returns information on the given structure's disk usage and availability. This call is useful in determining storage usage.

ACCEPTS IN AC1: Device designator, must be a designator for a structure. If the generic designator DSK: is given, the connected structure is assumed.

RETURNS +1: Always, with number of pages in use in AC1, and number of pages not in use in AC2.

TOPS-20 MONITOR CALLS
(GDSKC)

GDSKC ERROR MNEMONICS:

DEVX1: Invalid device designator

Returns the status of a device for user I/O. (Refer to Section 2.4 for the descriptions of the status bits.) This call requires that the device be opened.

Also, this call will not return the status of a device for monitor I/O. For example, if GDSTS is executed after a tape mark is written (a monitor I/O operation) the GDSTS call will return the status of the last user record written.

ACCEPTS IN AC1: JFN

RETURNS +1: Always, with device-dependent status bits in AC2, and device-dependent information in AC3. For magnetic tape, AC3 contains the positive count of number of hardware bytes actually transferred in the left half and zero in the right half. For the line printer, AC3 contains the last value of the page counter register, or -1 if there is no page counter register.

For TCP/IP, the return sequence for network-connection files is:

AC2: Connection state

 .TCNOT Connection not open
 .TCFIN Connection closed
 .TCSYA Connection openable
 .TCSYS Connection opening
 .TCSYN Connection open

AC3: Foreign host number (octal)

AC4: Foreign port number (octal)

The GDSTS call is a no-op for devices without device-dependent status bits.

The SDSTS monitor call can be used to set the status bits for a particular device.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(GDSTS)

GDSTS ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open

Returns the entry vector for the Record Management System (RMS).
(Refer to the RMS Manual for more information on the Record Management System.)

RESTRICTIONS: Requires RMS software.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with entry vector length in the left half and
the entry vector address in the right half of AC2.

The SDVEC monitor call can be used to set the entry vector for RMS.

The XGSEV% monitor call can be used to get an extended special entry
vector for RMS entry vectors in nonzero sections.

Generates an illegal instruction interrupt on error conditions below.

GDVEC ERROR MNEMONICS:

ILINS5: RMS facility is not available

Gets a save file, copying or mapping it into the process as
appropriate. It updates the monitor's data base for the process by
copying the entry vector and the list of program data vector addresses
(PDVAs) from the save file. (See the .POADD function of the PDVOP%
monitor call.)

This call can be executed for either sharable or nonsharable save
files that were created with the SSAVE or SAVE monitor call,
respectively. The file must not be open.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability
enabled.

ACCEPTS IN AC1: Process handle,, flag bits and a JFN.

TOPS-20 MONITOR CALLS
(GET)

AC2: Lowest process page number in left half, and highest process page number in right half; or the address of an argument block. If this AC contains page numbers, those page numbers control the parts of memory that are loaded when GT%ADR is on.

RETURNS +1: Always

The defined bits in AC1 are as follows:

| Bit | Symbol | Meaning |
|-------|--------|--|
| 19 | GT%ADR | Use the memory address limits given in AC2. If this bit is off, all existing pages of the file (according to its directory) are mapped. |
| 20 | GT%PRL | Preload the pages being mapped (move the pages immediately.) If this bit is off, the pages are read in from the disk when they are referenced. |
| 21 | GT%NOV | Do not overlay existing pages and do return an error. If this bit is off, existing pages will be overlaid. |
| 22 | GT%ARG | If this bit is on, AC2 contains the address of an argument block. |
| 24-35 | GT%JFN | JFN of the save file |

The format of the argument block follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .GFLAG | Flags that indicate how the rest of the argument block is to be used. |
| 1 | .GLOW | Number of the lowest page in the process into which a file page gets loaded. This page must be within the section specified by .GBASE. |
| 2 | .GHIGH | Number of the highest page in the process into which a file page gets loaded. This page must be within the section specified by .GBASE. |
| 3 | .GBASE | Number of the section into which the file pages are loaded. You can specify the section for single-section save files only; use of this word with a multiple-section save file causes an error. The file pages are loaded into this section of memory regardless of the section specified in the save file. |

TOPS-20 MONITOR CALLS
(GET)

The following flag bits are defined for use in .GFLAG:

| Bit | Symbol | Meaning |
|-----|---------|--|
| 0 | GT%LOW | .GLOW contains the number of the lowest page within the process to use. |
| 1 | GT%HGHI | .GHIGH contains the number of the highest page within the process to use. |
| 2 | GT%BAS | .GBASE contains the number of the section to use. |
| 3 | GT%CCH | Clear the system's program cache. (WHEEL or OPERATOR capability is required for use of this bit.) |
| 4 | GT%CSH | Place in cache the name of the program being loaded into memory. (WHEEL or OPERATOR capability is required for use of this bit.) |

When the GET call is executed for a sharable save file, pages from the file are mapped into pages in the process, and the previous contents of the process's page are overwritten. If the file contains data for only a portion of the process's page, the remainder of the page is zeroed. Pages of the process not used by the file are unchanged.

When the GET call is executed for a nonsharable save file, individual words of the file are written into the process. Since these files usually do not have words containing all zeros, a GET call executed for a nonsharable file never clears memory. The GET call never loads the accumulators.

The GET JSYS interacts with the JFN of the file that the GET is performed upon in the following ways:

1. If the GET is performed on a CSAVE file, a file on a non-disk device, or a file that has another JFN open on it, the JFN is released.
2. Under normal conditions for a file with only one JFN open on it, if the GET succeeds, it will eventually cause an implicit CLOSF for the file on which the GET was performed. This occurs through the following mechanism: GET changes the owner of the file from the process that issued the GET to the process into which the file is mapped. When the latter process is killed, the JFN is released.

Because a program can not be sure that GET has or has not released the JFN, the program should not attempt to release the JFN itself or attempt to use the JFN again (assuming that the GET actually succeeded). At the time that a program tried to erroneously release

TOPS-20 MONITOR CALLS
(GET)

the JFN itself, the JFN might be associated with a file other than the file on which the GET was performed. This can be a source of program errors that are difficult to trace.

This call can cause several software interrupts or process terminations on some file conditions.

A GET call performed on an execute-only process is illegal unless the process is .FHSLF. If the JFN specified in the GET call refers to a file for which the user only has execute-only access, then the process specified must be a virgin process.

Generates an illegal instruction interrupt on the following error conditions:

GET ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process
GETX1: Invalid save file format
GETX2: System Special Pages Table full
GETX3: Illegal to overlay existing pages
GETX4: Illegal to specify .GBASE for multisection file.
SSAVX1: Illegal to save files on this device
OPNX2: File does not exist

All file errors can occur.

Returns a word from the specified system table. (Refer to Section 2.3.2.)

ACCEPTS IN AC1: Index into table in the left half, and table number in the right half

RETURNS +1: Failure, error code in AC1

+2: Success, 36-bit word from the specified table in AC1

If -1 is given as the index, this call returns the negative of the length of the specified table.

The table number can be obtained with the SYSGT call. However, the recommended procedure is to use the symbol definition from the MONSYM file for the table number. (Refer to Chapter 2 for the system table definitions.)

TOPS-20 MONITOR CALLS
(GETAB)

The GETAB monitor call requires the process to have GETAB capability available, but not enabled (SC%GTB in the process capability word).

GETAB ERROR MNEMONICS:

GTABX1: Invalid table number
GTABX2: Invalid table index
GTABX3: GETAB privileges required

Returns the most recent error condition encountered in a process. The most recent error is always saved in the Process Storage Block.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with process handle in left half of AC2 and most recent error condition in right half of AC2.

The SETER monitor call can be used to set the most recent error condition encountered in a process.

GETER ERROR MNEMONICS:

LSTRX1: Process has not encountered any errors

Obtains information about the specified job.

RESTRICTONS: Requires SC%GTB capability in the process capability word.

ACCEPTS IN AC1: Job number, or -1 for current job, or 400000+TTY number

AC2: Negative of the length of the block in which to store the information in the left half, and the beginning address of the block in the right half

AC3: Word number (offset) of first entry desired from job information table

RETURNS +1: Failure, error code in AC1

+2: Success, with updated pointer in AC2 and requested entries stored in specified block; if the job does

TOPS-20 MONITOR CALLS
(GETJI)

not exist, returns +2, with -1 in Word 0 of the
specified block

When a terminal designator is given in AC1, the information returned is for the job running on that terminal.

The system begins copying the entries from the job information table, starting with the offset given in AC3, into the address specified in the right half of AC2. The number of entries copied is minus the number given in the left half of AC2, or is the number remaining in the table, whichever is smaller.

Because AC2 is updated on a successful return, it cannot be used for the returned data.

The format of the job information table is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .JIJNO | Job number |
| 1 | .JITNO | Job's terminal number (-1 means the job is detached) |
| 2 | .JIUNO | Job's user number |
| 3 | .JIDNO | Job's connected directory number |
| 4 | .JISNM | Subsystem name (SIXBIT) |
| 5 | .JIPNM | Program name (SIXBIT) |
| 6 | .JIRT | Run time (in milliseconds) |
| 7 | .JICPJ | Controlling PTY job number (-1 means the job is not controlled by a PTY) |
| 10 | .JIRTL | Run time limit (as set by the TIMER call) A zero means no time limit is in effect. |
| 11 | .JIBAT | Job is controlled by Batch, if -1 (as set by the MTOPR call) |
| 12 | .JIDEN | Default for magnetic tape density (as set by the SETJB call) |
| 13 | .JIPAR | Default for magnetic tape parity (as set by the SETJB call) |
| 14 | .JIDM | Default for magnetic tape data mode (as set by the SETJB call) |
| 15 | .JIRS | Default number for magnetic tape record size in bytes (as set by the SETJB call) |
| 16 | .JIDFS | Deferred spooling in effect, if 1 (as set by the SETJB call) |
| 17 | .JILNO | Job's logged-in directory number |
| 20 | .JISRM | Byte pointer to area to receive job's session remark. This pointer is supplied by the user before issuing the GETJI call. |
| 21 | .JILLN | The date and time of the user's last login before the user logged in the current job |
| 22 | .JISRT | Job CPU time at start of last session. To compute CPU time for this session, subtract .JISRT value from current job CPU time (.JIRT). |

TOPS-20 MONITOR CALLS
(GETJI)

| | | |
|----|--------|--|
| 23 | .JISCT | Console time at start of last session. To compute the console time for this session, subtract .JISCT value from current console time (obtainable with RUNTM monitor call). |
| 24 | .JIT20 | Indicates if job is at EXEC level or program level. (-1 = EXEC, 0 = program) |
| 25 | .JISTM | Returns time when job was created (when CTRL/C was performed). A -1 is returned if the system time and date were not set when the job started. |
| 26 | .JIBCH | Batch stream number and batch flags B0-1 OB%WTO Write-to-operator capabilities 0 .OBALL WTO (write-to-operator) and WTOR (write-to-operator with reply) 1 .OBNWR No WTOR allowed 2 .OBNOM No message allowed B10 OB%BSS Indicates that field OB%BSN (below) contains a batch-stream number B11-17 OB%BSN Batch-stream number |
| 27 | .JILLO | Logical location (node name). This word indicates the logical location of the job. This job location is typically used to cause output to be routed to a remote station, such as an IBM termination station or a DN200 remote station. |
| 30 | .JILJI | Local job index. Index into system-wide job tables. |
| 31 | .JIBSN | Batch sequence number. |
| 32 | .JIBJN | Batch job name. |
| 33 | .JIBID | Batch request ID. |
| 34 | .JICT | Job's connect time. |
| 35 | .JINLD | Job's last non-interactive login. |

The current highest GETJI offset is given by symbol .JIMAX.

GETJI ERROR MNEMONICS:

| | |
|---------|------------------------------|
| GTJIX1: | Invalid index |
| GTJIX2: | Invalid terminal line number |
| GTJIX3: | Invalid job number |
| GTJIX4: | No such job |

Returns the name of the program currently being used by the job. This name will have been declared previously with the SETNM or SETSN monitor call.

TOPS-20 MONITOR CALLS
(GETNM)

RETURNS +1: Always, with SIXBIT name of program in AC1

Requests access to the specified system resource from the installation's access-control program.

ACCEPTS IN AC1: Function code

AC2: Address of argument block (if required)

AC3: Length of the argument block (the maximum permissible length is specified by symbol .GOKMZ)

AC4: Job number or user number request is for

RETURNS +1: Always, with 0 in first word of status block if access granted

1B18 set to one + error number in first word of status block if request denied. An illegal instruction trap is generated.

Function Codes:

| Code | Symbol | Meaning |
|------|--------|--|
| 1 | .GOASD | Assign a device |
| | | Argument block (user-specified): |
| | | Word Symbol Contents |
| | | 0 .GEERB Error block address |
| | | 1 .GEADD Device designator |
| 2 | .GOCAP | Enable capabilities (right half privileges only) |
| | | Argument block (user-specified): |
| | | Word Symbol Contents |
| | | 0 .GEERB Error block address |
| | | 1 .GENCP New capability word |

TOPS-20 MONITOR CALLS
(GETOK%)

- 3 .GOCJB Allow CRJOB JSYS to be executed
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
- 4 .GOLOG Allow LOGIN
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
| 1 | .GELUN | User number |
- 5 .GOCFK Allow CFORK (only done after n'th fork). N is an installation-defined parameter specified by monitor symbol DGOFKN.
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------------------------|
| 0 | .GEERB | Error block address |
| 1 | .GEFCT | Number of forks already in use by job |
- 6 .GOTBR Set terminal baud rate
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|-----------------------------|
| 0 | .GEERB | Error block address |
| 1 | .GELIN | Line number |
| 2 | .GESPD | Input speed ,, Output speed |
- 7 .GOLGO Inform the access-control program of a logout.
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|--|
| 0 | .GEERB | Error block address |
| 1 | .GEUSD | Number of pages used |
| 2 | .GEQUO | Directory quota |
| 3 | .GERLG | Number of the job to be logged out, or -1 if the requesting job is to be logged out. |

TOPS-20 MONITOR CALLS
(GETOK%)

- 10 .GOENQ Allow setting of ENQ quota
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
| 1 | .GEEQU | Desired quota |
| 2 | .GEEUN | Job number |
-
- 11 .GOCRD Allow directory creation
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|--|
| 0 | .GEERB | Error block address |
| 1 | .GECFL | CRDIR% flags (this is the argument the user has passed into CRDIR% in AC 2) |
| 2 | .GEDIR | Block of 11 words containing STR:<DIRECTORY> |
| 15 | .GECAB | Block of 25 words containing the actual CRDIR% argument block. Note any byte pointers in the argument block are meaningless since they point to addresses in the user's own address space. |
-
- 12 .GOSMT Allow MOUNT of structure
- Must be given once to increment the mount count and once to decrement the mount count.
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
| 1 | .GESDE | Device designator |
-
- 13 .GOMDD Allow entry to MDDT
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
-
- 14 .GOCLS Set scheduler class for a job

TOPS-20 MONITOR CALLS
(GETOK%)

Argument block (user-specified):

| Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
| 1 | .GEJOB | Job number |
| 2 | .GECLS | Class desired |

15 .GOCL0 Set scheduler class at login

This function is executed by the monitor when a login occurs and class scheduling is enabled (but not by accounts). The access-control program must then determine which class to put the user in.

Argument block (user-specified):

| Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |

16 .GOMTA MT: access request

Argument block (user-specified):

| Word | Symbol | Contents |
|------|--------|------------------------------|
| 0 | .GEERB | Error block address |
| 1 | .GEACC | Access code from HDR1 label |
| 2 | .GEUSN | User number |
| 3 | .GEUNT | MT: unit number |
| 4 | .GEACD | Desired access bits (FP%xxx) |
| 5 | .GELTP | Label type (.LTxxx) |

17 .GOACC Allow ACCESS or CONNECT

Argument block (user-specified):

| Word | Symbol | Contents |
|------|--------|-----------------------|
| 0 | .GEERB | Error block address |
| 1 | .GOAC0 | Flags from ACCES JSYS |
| 2 | .GOAC1 | Directory number |

20 .GOOAD Allow device assignment due to OPENF

Argument block (user-specified):

| Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
| 1 | .GEADD | Device designator |

TOPS-20 MONITOR CALLS
(GETOK%)

- 21 .GODNA Allow access to DECNET
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
- 22 .GOANA Allow TCP/IP access
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | error block address |
- 23 .GOATJ Allow ATTACH
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|---------------------|
| 0 | .GEERB | Error block address |
| 1 | .GOTJB | Target job number |
| 2 | .GOTTY | Source TTY number |
- 24 .GOINF Allow INFO% execution
- Argument block (user-specified):
- | Word | Symbol | Contents |
|------|--------|-----------------------------------|
| 0 | .GEERB | Error block address |
| 1 | .GEJOB | Job number |
| 2 | .GECIN | CI node to execute INFO% function |
| 3 | .GEINF | INFO% function number |
- 25 .GOLAT Allow execution of LATOP% JSYS
- Argument block format (user-specified):
- | Word | Symbol | Contents |
|------|--------|---|
| 0 | .GEERB | Error block address |
| 1 | .GEJOB | Job number |
| 2 | .GEFUN | Flags,,Function Code Flags included here are LA%PSI, LA%QUE, LA%SYS, and LA%JOB. See LATOP% functions .LARHC and .LATHC for additional information. |

TOPS-20 MONITOR CALLS
(GETOK%)

| | | |
|----|--------|---|
| 3 | .GELTN | Four words, containing the ASCIZ node name (or 0) |
| 7 | .GELTP | Four words, containing the ASCIZ port name (or 0) |
| 11 | .GELTS | Four words, containing the ASCIZ service name (or 0) |
| 26 | .GOCTM | Allow incoming CTERM connection |
| | | Argument block format (user-specified): |
| | | Word Symbol Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GEWHO 13 (octal) words containing the string NODE::USER who is attempting the incoming CTERM connection. If the username is not easily determined, then the string will simply be the node. |
| 27 | .GOTTM | Allow use of TTMSG% monitor call |
| | | Argument block (user-specified): |
| | | Word Symbol Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GEDTY ACl as given to the TTMSG% JSYS |
| 30 | .GOSMN | Allow system parameters to be set with SMON% |
| | | Argument block (user-specified): |
| | | Word Symbol Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GESMF SMON% function number |
| | 2 | .GESMV New value for function |
| 31 | .GOHSY | Allow use of the HSYS% monitor call |
| | | Argument block (user-specified): |
| | | Word Symbol Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GESDT Shutdown time (internal format) |
| | 2 | .GERES System resume time (internal format) |

TOPS-20 MONITOR CALLS
(GETOK%)

| 32 | .GOSGT | <p>Allow access of information via SYSGT%</p> <p>Argument block (user-specified):</p> <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>.GEERB</td> <td>Error block address</td> </tr> <tr> <td>1</td> <td>.GETBN</td> <td>SIXBIT table name</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .GEERB | Error block address | 1 | .GETBN | SIXBIT table name | | | |
|------|--------|--|------|--------|----------|---|--------|---------------------|---|--------|--------------------------------|---|--------|--|
| Word | Symbol | Contents | | | | | | | | | | | | |
| 0 | .GEERB | Error block address | | | | | | | | | | | | |
| 1 | .GETBN | SIXBIT table name | | | | | | | | | | | | |
| 33 | .GOGTB | <p>Allow access of information via GETAB%</p> <p>Argument block (user-specified):</p> <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>.GEERB</td> <td>Error block address</td> </tr> <tr> <td>1</td> <td>.GETBN</td> <td>Index into table,,table number</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .GEERB | Error block address | 1 | .GETBN | Index into table,,table number | | | |
| Word | Symbol | Contents | | | | | | | | | | | | |
| 0 | .GEERB | Error block address | | | | | | | | | | | | |
| 1 | .GETBN | Index into table,,table number | | | | | | | | | | | | |
| 34 | .GOOPN | <p>Allow opening a file that is set secure</p> <p>Argument block (user-specified):</p> <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>.GEERB</td> <td>Error block address</td> </tr> <tr> <td>1</td> <td>.GEOAC</td> <td>AC 2 of OPENF%</td> </tr> <tr> <td>2</td> <td>.GEFIL</td> <td>226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being opened</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .GEERB | Error block address | 1 | .GEOAC | AC 2 of OPENF% | 2 | .GEFIL | 226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being opened |
| Word | Symbol | Contents | | | | | | | | | | | | |
| 0 | .GEERB | Error block address | | | | | | | | | | | | |
| 1 | .GEOAC | AC 2 of OPENF% | | | | | | | | | | | | |
| 2 | .GEFIL | 226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being opened | | | | | | | | | | | | |
| 35 | .GORNF | <p>Allow renaming a file that is set secure</p> <p>Argument block (user-specified):</p> <table border="0" style="margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>.GEERB</td> <td>Error block address</td> </tr> <tr> <td>1</td> <td>-----</td> <td>Not used</td> </tr> <tr> <td>2</td> <td>.GEFIL</td> <td>226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being renamed</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .GEERB | Error block address | 1 | ----- | Not used | 2 | .GEFIL | 226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being renamed |
| Word | Symbol | Contents | | | | | | | | | | | | |
| 0 | .GEERB | Error block address | | | | | | | | | | | | |
| 1 | ----- | Not used | | | | | | | | | | | | |
| 2 | .GEFIL | 226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being renamed | | | | | | | | | | | | |
| 36 | .GODLF | <p>Allow deleting a file that is set secure (either through DELF% or DELNF% monitor calls)</p> <p>Argument block (user-specified):</p> | | | | | | | | | | | | |

TOPS-20 MONITOR CALLS
(GETOK%)

| Word | Symbol | Contents |
|------|--------|--|
| | 0 | .GEERB Error block address |
| | 1 | .GEDAC Bits selected in user's AC 1 |
| | 2 | .GEFIL 226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being deleted |
| 37 | .GOTLK | Allow use of the TLINK% monitor call |
| | | Argument block (user-specified): |
| Word | Symbol | Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GETTB TLINK% flags,,object designator |
| | 2 | .GERMT Remote designator |
| 40 | .GOCRL | Allow use of the .CLNS1, .CLNSA or .CLNSY functions of the CRLNM% monitor call |
| | | Argument block (user-specified): |
| Word | Symbol | Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GECFN CRLNM% function |
| | 2 | .GELNM Block of 16 words that contain the logical name for .CLNS1 and .CLNSY functions |
| 41 | .GODTC | Inform access control job of DTACH% |
| | | Argument block (user-specified): |
| Word | Symbol | Contents |
| | 0 | .GEERB Error block address |
| 42 | .GOCFD | Allow CHFDB% to set or clear FB%SEC on a file |
| | | Argument block (user-specified): |
| Word | Symbol | Contents |
| | 0 | .GEERB Error block address |
| | 1 | .GESFS Contents of .FBCTL in file's FDB |
| | 2 | .GEFIL 226 (octal) words containing STR:<DIRECTORY>NAME.EXT.VER of file being deleted |

TOPS-20 MONITOR CALLS
(GETOK%)

| | | | | |
|----|--------|----------------|--------|-----------------------------|
| 43 | .GOGTD | GTDIR% JSYS | | |
| | | Argument block | | |
| | | Word | Symbol | Contents |
| | | 0 | .GEERB | Error block address |
| | | 1 | .GEDNO | Directory number |
| 44 | .GOSTD | STAD% JSYS | | |
| | | Argument block | | |
| | | Word | Symbol | Contents |
| | | 0 | .GEERB | Error block address |
| | | 1 | .GESTT | Time to set |
| 45 | .GODSK | DSKOP% JSYS | | |
| | | Argument block | | |
| | | Word | Symbol | Contents |
| | | 0 | .GEERB | Error block address |
| | | 1 | .GEST1 | User AC1 |
| | | 2 | .GEST2 | User AC2 |
| | | 3 | .GEST3 | User AC3 |
| | | 4 | .GEST4 | User AC4 |
| 46 | .GOSJP | SJPRI% JSYS | | |
| | | Argument block | | |
| | | Word | Symbol | Contents |
| | | 0 | .GEERB | Error block address |
| | | 1 | .GEST1 | User's AC1 (job number) |
| | | 2 | .GEST2 | User's AC2 (priority word) |
| 47 | .GOSPR | SPRIW% JSYS | | |
| | | Argument block | | |
| | | Word | Symbol | Contents |
| | | 0 | .GEERB | Error block address |
| | | 1 | .GEST1 | User's AC1 (process handle) |
| | | 2 | .GEST2 | User's AC2 (priority word) |

TOPS-20 MONITOR CALLS
(GETOK%)

400000+n Customer-reserved functions

The argument block (user-specified) has the same format as the error block format shown below. The contents of word 1 are ignored.

Error block format (returned):

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .GESIZ | Count of words in this block (including this word) |
| 1 | .GEERN | Error Number |
| 2 | .GEPTR | Byte pointer to error string location |
| 3 | .GEBSZ | Maximum bytes user can accept in error string |

The format of the status block for user-defined functions will depend on the design of the particular access-control program.

The user supplies all arguments in the argument block. In the error block, the user supplies words 0, 2, and 3. If an error string is provided by the program doing the GIVOK%, then the byte pointer and count are updated. If the user is not interested in the reason for the rejection, the address of the error block can be 0. If the error block is less than 4 words, only the available words will be used. If the byte pointer is 0, no string will be returned.

Error codes are of the form 1B18+n. They are not standard TOPS-20 error codes and therefore cannot be given to ERSTR to produce a string. The access-control program must supply a string if one is needed.

Generates an illegal instruction interrupt on the following error conditions:

GETOK% ERROR MNEMONICS:

| | |
|---------|---|
| ARGX04: | Argument block too small |
| ARGX05: | Argument block too long |
| ARGX26: | File is off line |
| MONX01: | Insufficient system resources |
| GOKER1: | Illegal function |
| GOKER2: | Request denied by Access Control Facility |

TOPS-20 MONITOR CALLS
(GEVEC)

Returns the section-relative entry vector of the specified process. (See Section 2.7.3.) The process must be one that runs in a single section of memory. See the XGVEC% monitor call to obtain the entry vector of a multisection program.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with specified process's entry vector word in AC2

The SEVEC monitor call can be used to set the process's entry vector. (See the PDVOP% monitor call for a description of the program data vector.)

Generates an illegal instruction interrupt on the following error conditions:

GEVEC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

Gets a handle on a process that currently is not known to the caller but is known to another process. The handle returned can then be used by the caller to refer to the process of interest.

ACCEPTS IN AC1: Handle of the process that has a handle on the process of interest

AC2: Process handle, used by the process named in AC1, that refers to the process of interest. This handle must be a relative handle (in the range 400000 to 400777) and must refer to an existing process.

RETURNS +1: Failure, with error code in AC1.

+2: Success, with a handle in AC1 that is usable by the caller to refer to the desired process. This handle is not the same as the one given in AC2 (is different from the one used by the process in AC1 to refer to the desired process).

TOPS-20 MONITOR CALLS
(GFRKH)

Generates an illegal instruction interrupt on error conditions below.

GFRKH ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle
FRKH6: All relative process handles in use
GFRK1: Invalid process handle

Returns the process structure of the current job from a given process downward.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Process handle of the starting point

AC2: B0(GF%GFH) Return relative process handles for each process

B1(GF%GFS) Return status for each process

AC3: The left half contains the negative of the number of words in the block in which to store the process structure, and the right half contains the address of the first word of the block

RETURNS +1: Failure, error code in AC1

+2: Success, all process handles are returned

The handle of the current process is always returned as .FHSLF regardless of the setting of GF%GFH. Any user can specify a process handle of .FHTOP (causing GFRKS to start with the top level process). However, the user must have WHEEL or OPERATOR capability enabled to specify .FHTOP, set GF%GFH and receive relative handles for all processes from .FHTOP on down. Otherwise, only process handles that the issuing process is entitled to receive will be returned. Also, if the request will cause the monitor to exceed the per-process fork handle limit, only that number of handles that will fit within the limit will be returned.

TOPS-20 MONITOR CALLS
(GFRKS)

Table Format

```

=====
!           !           !           !
3 words    !   parallel !   inferior !
per entry  !   pointer  !   pointer  !
!           !           !           !
=====
!           !           !           !
!   superior ! process handle !
!   pointer  ! or 0 if GF%GFH !
!           ! was off, or when no !
!           ! more process handles !
!           ! are left for the !
!           ! process !
!           !           !
=====
!           !           !           !
This word  !           status word !
is -1 if  !           !
GF%GFS    !           !
is off.   !           !
!           !           !
=====

```

NOTE

Pointers in table are memory addresses of other table entries, or 0 if no such structure exists.

The execution of the GFRKS call terminates before the entire process structure has been returned if the block in which to store the structure information is too small. If this happens, this call returns as much of the structure as can fit in the block, then generates an error message. If all process handles are in use, this call returns the entire structure, but the extra handles will not be assigned (will be zero).

Generates an illegal instruction interrupt on error conditions below.

GFRKS ERROR MNEMONICS:

- FRKH1: Invalid process handle
- FRKH2: Illegal to manipulate a superior process
- FRKH3: Invalid use of multiple process handle
- FRKH6: All relative process handles in use
- GFKSX1: Area too small to hold process structure

TOPS-20 MONITOR CALLS
(GFUST)

Returns the name of either the author of the file or the user who last wrote to the file.

ACCEPTS IN AC1: Function code in the left half, and JFN of the file in the right half

AC2: Pointer to the string in which to store the name

RETURNS +1: Always, with an updated string pointer in AC2

The defined functions are as follows:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .GFAUT | Return the name of the author of the file. |
| 1 | .GFLWR | Return the name of the user who last wrote to the file. |

The SFUST monitor call can be used to set the name of either the author of the file or the user who last wrote to the file.

Generates an illegal instruction interrupt on error conditions below.

GFUST ERROR MNEMONICS:

GFUSX1: Invalid function
GFUSX2: Insufficient system resources
GFUSX3: File expunged
GFUSX4: Internal format of directory is incorrect
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DESX8: File is not on disk
DESX10: Structure is dismounted
DELFX6: Internal format of directory is incorrect
DIRX2: Insufficient system resources
DIRX3: Internal format of directory is incorrect

TOPS-20 MONITOR CALLS
(GIVOK%)

Allows a privileged access-control program (written by the installation) to allow or disallow a user program's access to a specified system resource.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Request number (from RCVOK% message)

AC2: 0 = request granted
1B18 + error number = request denied

AC3: Pointer to ASCIZ string (maximum of 80 characters) or 0. This string is an error message or information message to be returned to the user.

RETURNS +1: Always

Generates an illegal instruction interrupt on error conditions below.

GIVOK% ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required
GOKER3: JSYS not executed within ACJ fork

Returns information pertaining to the current job.

RETURNS +1: Always, with

AC1 Containing the user number under which the job is running.

AC2 Containing the directory number to which the job is connected.

AC3 Containing the job number.

AC4 Containing the terminal number attached to the job, or -1 if no terminal is attached to job.

TOPS-20 MONITOR CALLS
(GNJFN)

Assigns the JFN to the next file in a group of files that have been specified with wildcard characters. The next file in the group is determined by searching structures and directories in the order described in Section 2.2.3. The flags returned from the GTJFN call are given to the GNJFN call as an argument to indicate the fields of the file specification that contain wildcard characters.

ACCEPTS IN AC1: Indexable file handle returned by GTJFN flags returned by GTJFN in the left half and the JFN in the right half)

RETURNS +1: Failure, including no more files in the group. JFN is released if there are no more files in the group. This return occurs on the first call to GNJFN if no flags indicating wildcard fields are on in the left half of AC1.

+2: Success, same JFN is assigned to the next file in the group. The following flags are set (if appropriate) in the left half of AC1:

| | | |
|-----|--------|-------------------|
| B13 | GN%STR | structure changed |
| B14 | GN%DIR | directory changed |
| B15 | GN%NAM | name changed |
| B16 | GN%EXT | file type changed |

The GNJFN call uses the flags returned in the left half of AC1 on a GTJFN call to determine the fields containing wildcards and the default generation number. Note that the GNJFN call returns a different set of flags in the left half of AC1 than the GTJFN call returns. Because all calls to GNJFN should use the flags originally returned by GTJFN, programs must save the returned GTJFN flags for use in the GNJFN call.

The file currently associated with the JFN must be closed when the GNJFN call is executed. The indexable file handle for a file that has been renamed cannot be used as an argument to GNJFN.

GNJFN will not find invisible files unless bit G1%IIN was set in the GTJFN call.

GNJFN ERROR MNEMONICS:

| | |
|---------|---|
| DESX1: | Invalid source/destination designator |
| DESX2: | Terminal is not available to this job |
| DESX3: | JFN is not assigned |
| DESX4: | Invalid use of terminal designator or string pointer |
| GNJFX1: | No more files in this specification |
| GNJFX2: | Could not step to next file because current file no longer exists |

TOPS-20 MONITOR CALLS
(GNJFN)

OPNX1: File is already open
STRX09: Prior structure mount required

Returns the primary JFNs of the specified process.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with primary input JFN in the left half of AC2, and the primary output JFN in the right half of AC2. Unless the primary JFNs have been reset, AC2 contains -1 (777777,,777777), indicating TTY: as the primary I/O source/destination.

The SPJFN monitor call can be used to set the primary JFNs.

Generates an illegal instruction interrupt on error conditions below.

GPJFN ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

Returns the current date in the internal system format. (See Section 2.9.2.)

RETURNS +1: Always, with day in the left half of AC1, and fraction of day in right half of AC1

If the system does not have the current date set, AC1 contains -1.

The STAD monitor call can be used to set the system's date.

TOPS-20 MONITOR CALLS
(GTDAL)

Returns the disk allocation for the specified directory.

ACCEPTS IN AC1: Directory number (-1 indicates the connected directory)

RETURNS +1: Always, with

AC1 Containing the working disk storage limit (logged-in quota) for the directory.

AC2 Containing the number of pages being used.

AC3 Containing the permanent disk storage limit (logged-out quota) for the directory.

Generates an illegal instruction interrupt on error conditions below.

GTDAL ERROR MNEMONICS:

DIRX1: Invalid directory number

DELFX6: Internal format of directory is incorrect

STRX10: Structure is offline

Returns information about the given directory.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Directory number (36-bit)

AC2: Address of argument block in caller's address space in which to return the directory information

AC3: Byte pointer to the password string

RETURNS +1: Always, with updated byte pointer in AC3

The argument block returned to the caller has the same format as the CRDIR call's argument block. Word zero (.CDLEN) of the argument block must contain the length of the argument block in which GTDIR is to store the directory information being returned. If this word is zero,

TOPS-20 MONITOR CALLS
(GTDIR)

GTDIR assumes the length of the argument block is 15 (octal) words long, and returns only 15 (octal) words.

The password of the directory must be placed in the string to which AC3 points. Word 1(.CDPSW) of the returned argument block also points to this string.

The count of words to be returned in the user group list is given in word 14 (.CDDGP) of the argument block. This count must be one more than the number of words to be returned in the group list. This is because GTDIR returns a zero word as the last word in the group list.

If the directory number given is zero, the GTDIR monitor call returns the system default settings for the following directory parameters:

```
working disk storage quota (.CDLIQ)
permanent disk storage quota (.CDLOQ)
default file protection (.CDFPT)
default directory protection (.CDDPT)
default file retention count (.CDRET)
maximum number of subdirectories allowed (.CSDSQ)
online expiration period (.CDDNE)
offline expiration period (.CDDFE)
date and time of last interactive login (.CDLLD)
date and time of last non-interactive login (.CDNLD)
count of failed logins, RH: interactive, LH: non-interactive
(.CDFPA)
```

Either one of the following conditions must be satisfied for the caller to obtain all information (including the password) about the given directory:

1. The caller has WHEEL or OPERATOR capability enabled.
2. The caller has owner access to the directory.

Note that if password encryption is enabled, the returned password will be encrypted. To obtain all other information (other than the password) of the given directory, the caller must have at least owner access to the directory. (See Section 2.2.6 for a description of owner access.)

Generates an illegal instruction interrupt on error conditions below.

GTDIR ERROR MNEMONICS:

```
GTDIX1:  WHEEL or OPERATOR capability required
GTDIX2:  Invalid directory number
MSTX32:  Structure was not mounted
STRX10:  Structure is offline
```

TOPS-20 MONITOR CALLS
(GTFDB)

Returns some or all of the file descriptor block for the specified file. (See Section 2.2.8 for the format of this block.)

ACCEPTS IN AC1: JFN

AC2: Number of words to be read in the left half and the word number (offset) of the first entry desired from the file descriptor block in the right half.

AC3: Address in caller's address space for storing the data returned

RETURNS +1: Always

The following instruction will set up AC2 for reading the entire FDB:

```
HRLZI AC2,.FBLEN
```

The program receives an error (GFDBX2) if it requests more words than there are words remaining in the FDB. For TOPS-20 V4, the size of the FDB has been increased. If the left half of AC2 contains the current maximum size of the FDB (.FBLEN), but the FDB is an older, small FDB, then the extra words will contain zeroes.

See Section 2.2.8 for the various JSYSs used to modify the FDB.

Generates an illegal instruction interrupt on error conditions below.

GTFDB ERROR MNEMONICS:

GFDBX1: Invalid displacement
GFDBX2: Invalid number of words
GFDBX3: List access required
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
STRX10: Structure is offline

Obtains information about TCP/IP hosts.

TOPS-20 MONITOR CALLS
(GTHST%)

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: Function code.
The following bits are defined to be supplied in AC1 with the function code:

1B14(GH%QCL) Class argument supplied (functions .GTHMX,.GTHVN, .GTHOS only). If not specified, the class for a DNS query is assumed to be Internet.

1B16(GH%STA) Return status code in AC1 on success or partial success. If this bit is not set, only total success will result in a successful return. The codes that are returned are:

| Value | Symbol Meaning |
|-------|--|
| 0 | .GTHVS Total success. |
| 1 | .GTHVF Not found in namespace (authoritative). This value is returned instead of GTHSX8. |
| 2 | .GTHVT Timeout while waiting for name server response. This value is returned instead of GTHSX7 (non-authoritative) or GTHSX4. |

AC2: Function-specific argument

AC3: Function-specific argument

AC4: Function-specific argument

RETURNS +1: Failure, error code in AC1
+2: Success, function-specific data returned in AC's

| Function | Symbol | Meaning |
|----------|--------|--|
| 0 | .GTHSZ | Returns negative number of host names, negative length of HSTSTS table, and local host number. |

User-supplied arguments:

None

Returned data:

AC2: -number host names,,0
AC3: -length of HSTSTS table,,0
AC4: local host number (in 32-bit Internet format)

TOPS-20 MONITOR CALLS
(GTHST%)

- 1 .GTHIX Returns the name string associated with the host, the host number, and the host status. If the name returned is a nickname, HS%NCK is on in the status word.

 User-supplied arguments:

 AC2: destination byte pointer
 AC3: index into name table (returned by GETAB)

 Returned data:

 AC2: updated byte pointer
 AC3: host number
 AC4: host status
- 2 .GTHNS Returns the primary name for the given host number.

 User-supplied arguments:

 AC2: destination byte pointer
 AC3: host number

 Returned data:

 AC2: updated byte pointer
 AC3: host number
 AC4: host status
- 3 .GTHSN Translates the specified host name string to its host number. If the name specified is a nickname, HS%NCK will be on in the status word.

 User-supplied arguments:

 AC2: source byte pointer

 Returned data:

 AC2: updated byte pointer
 AC3: host number
 AC4: host status
- 4 .GTHHN Returns the current status of the given host.

 User-supplied arguments:

 AC3: host number

TOPS-20 MONITOR CALLS
(GTHST%)

Returned data:

AC3: host number
AC4: host status

- 5 .GTHHI Returns the host number and status of the host having the specified index into the host status table.

User-supplied arguments:

AC3: index into HSTSTS (returned by GETAB)

Returned data:

AC3: host number
AC4: host status

- 6 .GTHLN Returns the host number of this host on an Internet network.

User-supplied arguments:

AC2: network number, or host number of a network

Returned data:

AC3: host number on specified network

- 7 .GTHNT Returns status table of an Internet network.

User-supplied arguments:

AC2: network number, or host number of a network

AC3: address to store data

AC4: length, ,offset

- 10 .GTHLA Returns address of network interfaces.

User-supplied arguments:

AC3: address to store data

AC4: count of words available

Returned data:

AC4: list of all addresses host has (actual count of words)

TOPS-20 MONITOR CALLS
(GTHST%)

14 .GTHPN Translates the specified host name string to its host number. The host's primary name and IP address is returned.

User-supplied arguments:

AC2: source designator to host name string

AC4: destination designator for primary name string

Returned data:

AC2: updated source designator

AC3: primary host number

AC4: updated destination designator

15 .GTHMX Return mail exchange data. This data is intended for use only by programs wishing to deliver mail.

User-supplied arguments:

AC2: source designator to name for query

AC3: destination byte pointer to name block

AC4: address of argument block

Returned data:

AC2: updated source designator

AC3: updated byte pointer

Format of argument block

Word Symbol Meaning

| | | |
|---|--------|--|
| 0 | .GTHLN | On call, length of argument block in words including this word. On return, number of words returned including this word. |
|---|--------|--|

| | | |
|---|--------|---|
| 1 | .GHTHC | On call, class for MX records if GH%QCL is on in AC1. |
|---|--------|---|

| | | |
|---|--------|---|
| 2 | .GTHBC | On call, length of name block (pointed to by AC3) in bytes. On return, remaining length of buffer in bytes. |
|---|--------|---|

TOPS-20 MONITOR CALLS
(GTHST%)

3 .GTHNM On return, the pointer to the first mail exchange name. Words after this one contain pointers to the remaining mail exchange names. Each returned word is a byte pointer into the name block of a null terminated ASCII string.

16 .GTHAA Authenticate address. This function checks to see if an address is among those associated with the specified name. This is the right way to validate the host name associated with an open network connection. A success return indicates that the address was authenticated.

 User-supplied arguments:

 AC2: source designator to host name string

 AC3: address of host or -1 for local host

 Returned data:

 AC2: updated source designator

20 .GTHVN Validate name. This function checks to see if a name is found in one or more DNS resource records (RRs).

 User-supplied arguments:

 AC2: source designator for name to be validated

 AC3: LH: DNS class to match (if GH%QCL is on in AC1)
 RH: DNS type to match

 AC4: destination designator for canonical name

 Value Symbol DNS class

 1 .GTHCI Internet class

 Value Symbol DNS type

 1 .GHTHA A host address (type A RR)

 2 .GHTHN An authoritative name server (type NS RR)

 5 .GHTHC A canonical name (type CNAME RR)

TOPS-20 MONITOR CALLS
(GTHST%)

6 .GTHTS Start of a zone of authority (type SOA RR)

13 .GTHTW Well known service description (type WKS RR)

14 .GTHTP A domain name pointer (type PTR RR)

16 .GTHTH Host information (type HINFO RR)

17 .GTHTM Mail exchange (type MX RR)

200001 .GTHVH Validate host (match on any type A, MX, WKS, or HINFO RRs)

200002 .GTHVZ Validate zone (match on any type SOA or NS RRs)

Returned data:

AC2: updated source designator

AC3: class,,type pair that matched

AC4: updated destination designator

23 .GTHOS Operating system. Extracts the operating system name as a string from the DNS HINFO RR for a host name.

User-supplied arguments:

AC2: source designator for host name

AC3: destination designator for operating system name

AC4: class (if GD%QCL is on in AC1)

Returned data:

AC2: updated source pointer

AC3: updated destination pointer

24 .GTHDN Get DNS nameserver host information. This function is intended primarily for SYSDPY.

User-supplied arguments:

AC2: Index into DNS host table, starting at 0

TOPS-20 MONITOR CALLS
(GTHST%)

AC3: Address of four word block to store data

AC4: Number of words to return (1-4)

Returned data in argument block:

| Word | Symbol | Meaning |
|------|--------|---------------------------------|
| 0 | .GTHDA | IP address of DNS host |
| 1 | .GTHDT | Timeout in seconds for DNS host |
| 2 | .GTHDS | Success count for DNS host |
| 3 | .GTHDF | Failure count for DNS host |

Flags in host status word:

| Bits | Symbol | Meaning |
|-------|--------|--|
| 1B0 | HS%UP | Host is up |
| 1B1 | HS%VAL | Valid status |
| 7B4 | HS%DAY | Day when up if currently down |
| 37B9 | HS%HR | Hour |
| 17B13 | HS%MIN | 5 minute interval |
| 17B17 | HS%RSN | Reason |
| 1B18 | HS%SRV | Host is server |
| 1B19 | HS%USR | Host is user |
| 1B20 | HS%NCK | Nickname |
| 77B26 | HS%STY | System type mask |
| 1B27 | HS%NEW | RAS, RAR, RAP, etc |
| 1B29 | HS%SLF | Host is an alias |
| 1B30 | HS%NET | Host is a network name |
| 1B31 | HS%GAT | Host is a gateway |
| 1B32 | HS%DNS | Host name added from DNS data (as opposed to SYSTEM:HOSTS.TXT) |
| 1B33 | HS%INA | DNS PTR RR data was used (implies non-authoritative data) |
| 1B34 | HS%AUT | Authoritative answer from nameserver |

System Type Flags (HS%STY):

| Bits | Symbol | Meaning |
|------|--------|---------|
| 1B26 | .HS10X | TENEX |
| 2B26 | .HSITS | ITS |
| 3B26 | .HSDEC | TOPS-10 |
| 4B26 | .HSTIP | TIP |
| 5B26 | .HSMTP | MTIP |

TOPS-20 MONITOR CALLS
(GTHST%)

| | | |
|-------|--------|----------|
| 6B26 | .HSELF | ELF |
| 7B26 | .HSANT | ANTS |
| 10B26 | .HSMLT | MULTICS |
| 11B26 | .HST20 | TOPS-20 |
| 12B26 | .HSUNX | UNIX |
| 13B26 | .HSNET | Network |
| 14B26 | .HSFUZ | Fuzzball |
| 15B26 | .HSVMS | VMS |
| 16B26 | .HSTAC | TAC |
| 17B26 | .HSDOS | MSDOS |

GTHST% ERROR MNEMONICS:

| | |
|---------|--|
| GTHSX1: | No DNS name servers configured |
| GTHSX2: | Unknown host number |
| GTHSX3: | Unknown host name |
| GTHSX4: | Format error in DNS message |
| GTHSX5: | No interface to specified network |
| GTHSX6: | Invalid class for function |
| GTHSX7: | Server failed to find data (non-authoritative) |
| GTHSX8: | Data not found in namespace (authoritative) |
| GTHSX9: | String argument is too long |
| GTHX10: | System host tables full |
| GTJIX1: | Invalid index |
| ARGX02: | Invalid function |
| ARGX04: | Argument block too small |
| ARGX24: | Invalid count |

Returns a JFN for the specified file. Accepts the specification for the file from a string in memory or from a file, but not from both.

ACCEPTS IN AC1: GJ%SHT plus other flag bits in the left half, and default generation number in the right half

AC2: Source designator from which to obtain the file specification. (See flag bit GJ%FNS for specific values.)

RETURNS +1: Failure, error code in AC1

+2: Success, flags in the left half of AC1, and the JFN assigned in the right half of AC1. (This word is

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

called an indexable file handle and is given to the GNJFN call as an argument.) Updated string pointer in AC2, if pertinent.

All I/O errors can occur. These errors cause software interrupts or process terminations, and only a single return (+1) is given.

The string can represent the complete specification for the file:

dev:<directory>name.typ.gen/attributes

For parse-only JFNs, the file specification is also allowed to be

node::dev:<directory>name.typ.gen/attributes

One or more fields of the specification can be defined by a logical name. (See Section 2.2.2.) If any fields are omitted from the specification, the system will provide the values shown below.

| | |
|-----------|---------------------|
| device | connected structure |
| directory | connected directory |

NOTE

If neither device nor directory is specified, the default is DSK:, not the user's connected directory. If either device or directory is specified, the other is the user's connected structure/directory.

| | |
|------------|---|
| name | no default; this field must be specified |
| type | null |
| generation | highest existing number if the file is an input file. Next higher number if the file is an output file. |
| protection | protection of the next lower generation or for new files, protection as specified in the directory. |
| account | account specified when user logged in, unless changed by the CACCT or SACTF call. |

The JFNS monitor call can be used to obtain the file specification string associated with a given JFN. The flag bits that can be specified in AC1 are described as follows.

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

GTJFN Flag Bits

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | GJ%FOU | The file given is to be assigned the next higher generation number. This bit indicates that a new version of a file is to be created, and is usually set if the file is for output use. |
| 1 | GJ%NEW | The file specification given must not refer to an existing file (the file must be a new file). This bit has no effect on a parse-only JFN. |
| 2 | GJ%OLD | The file specification given must refer to an existing file. This bit has no effect on a parse-only JFN. |
| 3 | GJ%MSG | One of the appropriate messages is to be printed after the file specification is obtained, if the system is performing recognition on the file specification and the user ends his input by typing an ESC. !NEW FILE! !NEW GENERATION! !OLD GENERATION! !OK! if GJ%CFM (bit 4) is off !CONFIRM! if GJ%CFM (bit 4) is on |
| 4 | GJ%CFM | Confirmation from the user will be required (if GJ%FNS is on) to verify that the file specification obtained is correct. (See below for the valid confirmation characters.) |
| 5 | GJ%TMP | The file specified is to be a temporary file. |
| 6 | GJ%NS | Only the first specification in a multiple logical name assignment is to be searched for the file (do not search beyond the first name in a multiple logical name assignment). |
| 7 | GJ%ACC | The JFN specified is not to be accessed by inferior processes in this job. However, another process can access the file by acquiring a different JFN. To prevent the file from being accessed by other processes, the user's program should set OF%RTD(B29) in the OPENF call. |
| 8 | GJ%DEL | Files marked as deleted are to be considered by the system when it is searching for a file to assign to the JFN. |

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

- | | | |
|------|--------|---|
| 9-10 | GJ%JFN | These bits are off in the short form of the GTJFN call. |
| 11 | GJ%IFG | The file specification given is allowed to have one or more of its fields specified with a wildcard character (* or %). This bit is used to process a group of files and is generally used for input files. The monitor verifies that at least one value exists for each field that contains a wildcard and assigns the JFN to the first file in the group. The monitor also verifies that fields not containing wildcards represent a new or old file according to the setting of GJ%NEW and GJ%OLD. The GNJFN call can then be used to obtain the next file in the group. (See Section 2.2.3 for more information on wildcard characters in file specifications.) |
| 12 | GJ%OFG | The JFN is to be associated with the given file specification string only and not to the actual file. The string may contain wildcard characters (* or %) in one or more of its fields. It is checked for correct punctuation between fields, but is not checked for the validity of any field. This bit allows a JFN to be associated with a file specification even if the file specification does not refer to an actual file. The JFN returned cannot be used to refer to an actual file (for example, cannot be used in an OPENF call) but can be used to obtain the original input string (via JFNS). The fields in this string can then be used in a GTJFN-long form call as program defaults. However, if the original string contains the temporary file attribute (;T), this attribute is not "remembered" and thus is not returned on the JFNS call even though the bit indicating temporary status (JS%TMP) is set. All other fields (including the protection and account fields) can be returned by JFNS. |

When both B11(GJ%IFG) and B12(GJ%OFG) are on, the GTJFN call parses the specification given, verifying the existence of each field. When a wildcard character appears in a field, the GTJFN call checks the remaining fields for correct punctuation and returns a JFN for the file specification string only. That is, once a wildcard character is seen, the action taken is identical to that taken when only B12(GJ%OFG) is set. If no wildcard character appears in the string, the action is the same as if both bits were off.

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

| | | |
|-------|--------|---|
| 13 | GJ%FLG | Flags are to be returned in the left half of AC1 on a successful return. |
| 14 | GJ%PHY | Job-wide logical names (those defined by the user) are to be ignored by the monitor for this call. |
| 15 | GJ%XTN | This bit is off in the short form of the GTJFN call. |
| 16 | GJ%FNS | The contents of AC2 are to be interpreted as follows: 1. If this bit is on, AC2 contains an input JFN in the left half and an output JFN in the right half. The input JFN is used to obtain the file specification to be associated with the JFN. The output JFN is used to indicate the destination for printing the names of any fields being recognized. To omit either JFN, specify .NULIO (377777). 2. If this bit is off, AC2 contains a byte pointer to an ASCIZ string in memory that specifies the file to be associated with the JFN. |
| 17 | GJ%SHT | This bit must be on for the short form of the GTJFN call. |
| 18-35 | | The generation number of the file (between 1 and 377777) or one of the following: 0(.GJDEF) to indicate that the next higher generation number of the file is to be used if GJ%FOU (bit 0) is on, or to indicate that the highest existing generation number of the file is to be used if GJ%FOU is off. (This value is usually used in this field.) -1(.GJNHG) to indicate that the next higher generation number of the file is to be used if no generation number is supplied. -2(.GJLEG) to indicate that the lowest existing generation number of the file is to be used. -3(.GJALL) to indicate that all generation numbers (*) of the file are to be |

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

used and that the JFN is to be assigned to the first file in the group. (Bit GJ%IFG must be set.)

The GTJFN monitor call always reads the terminating character after the file specification string. (This character can be obtained by executing the BKJFN call followed by a BIN call.) The valid terminating characters are:

| | |
|------------------------|-------------------|
| line feed | left parenthesis |
| CTRL/L | right parenthesis |
| CTRL/Z | plus sign |
| carriage return | comma |
| exclamation point | slash |
| double quotation marks | equals sign |
| number sign | at sign (@) |
| ampersand | space |
| single quotation mark | ESC |

All of these characters except for ESC are also confirmation characters (see bit GJ%CFM above) and are called confirming terminators. If a confirming terminator is typed after the string, a confirmation message will not be typed to the user nor will the user be required to confirm the string obtained, regardless of the setting of GJ%MSG and GJ%CFM. On a successful return, the following flags are returned in the left half of AC1 if flag bit GJ%IFG, GJ%OFG, or GJ%FLG was on in the call.

Bits Returned on Successful GTJFN Call

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | GJ%DEV | The device field of the file specification contained wildcard characters. |
| 1 | GJ%UNT | The unit field of the file specification contained wildcard characters. This bit will never be set because wildcard characters are not allowed in unit fields. |
| 2 | GJ%DIR | The directory field of the file specification contained wildcard characters. |
| 3 | GJ%NAM | The filename field of the file specification contained wildcard characters. |

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

| | | |
|----|--------|---|
| 4 | GJ%EXT | The file type field of the file specification contained wildcard characters. |
| 5 | GJ%VER | The generation number field of the file specification contained wildcard characters. |
| 6 | GJ%UHV | The file used has the highest generation number because a generation number of 0 was given in the call. |
| 7 | GJ%NHV | The file used has the next higher generation number because a generation number of 0 or -1 was given in the call. |
| 8 | GJ%ULV | The file used has the lowest generation number because a generation number of -2 was given in the call. |
| 9 | GJ%PRO | The protection field of the file specification was given. |
| 10 | GJ%ACT | The account field of the file specification was given. |
| 11 | GJ%TFS | The file specification is for a temporary file. |
| 12 | GJ%GND | Files marked for deletion were not considered when assigning JFNs. This bit is set if GJ%DEL was not set in the call. |
| 13 | GJ%NOD | The node name field of the file specification was given. |
| 17 | GJ%INV | Invisible files were not considered when assigning JFNs. This bit is always on for the short form GTJFN. |

GTJFN ERROR MNEMONICS:

GJFX1: Desired JFN invalid
GJFX2: Desired JFN not available
GJFX3: No JFNs available
GJFX4: Invalid character in filename
GJFX5: Field cannot be longer than 39 characters
GJFX6: Device field not in a valid position
GJFX7: Directory field not in a valid position
GJFX8: Directory terminating delimiter is not preceded by a valid beginning delimiter

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

GJFX9: More than one name field is not allowed
GJFX10: Generation number is not numeric
GJFX11: More than one generation number field is not allowed
GJFX12: More than one account field is not allowed
GJFX13: More than one protection field is not allowed
GJFX14: Invalid protection
GJFX15: Invalid confirmation character
GJFX16: No such device
GJFX17: No such directory name
GJFX18: No such filename
GJFX19: No such file type
GJFX20: No such generation number
GJFX21: File was expunged
GJFX22: Insufficient system resources (Job Storage Block full)
GJFX23: Exceeded maximum number of files per directory
GJFX24: File not found
GJFX27: File already exists (new file required)
GJFX28: Device is not on-line
GJFX30: Account is not numeric
GJFX31: Invalid wildcard designator
GJFX32: No files match this specification
GJFX33: Filename was not specified
GJFX34: Invalid character "?" in file specification
GJFX35: Directory access privileges required
GJFX36: Internal format of directory is incorrect
GJFX37: Input deleted
GJFX38: File not found because output-only device was specified
GJFX39: Logical name loop detected
GJFX40: Undefined attribute in file specification
GJFX41: File name must not exceed 6 characters
GJFX42: File type must not exceed 3 characters
GJFX43: More than one ;T specification is not allowed
GJFX44: Account string does not match
GJFX45: Illegal to request multiple specifications for the same attribute

GJFX46: Attribute value is required
GJFX47: Attribute does not take a value
GJFX48: GTJFN input buffer is empty
GJFX49: Invalid attribute for this device
GJFX51: Byte count too small
GJFX55: Illegal to use node name
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged
DESX9: Invalid operation for this device
STRX09: Prior structure mount required
STRX10: Structure is offline
TCPXX1: No IP free space for TCB
TCPXX2: Unable to decode local side TCP of specification
TCPXX3: Unable to decode foreign side TCP of specification
TCPXX4: Generation found in TCP specification

TOPS-20 MONITOR CALLS
(GTJFN Short Form)

TCPXX5: TCP specification attribute not known to TCP
TCPXX6: Unable to decode CONNECTION attribute in TCP specification
TCPXX7: Unable to decode FOREIGN-HOST attribute in TCP specification
TCPXX8: Unable to decode LOCAL-HOST attribute in TCP specification
TCPXX9: Unable to decode PERSIST attribute in TCP specification
TCPX10: Unable to decode TIMEOUT attribute in TCP specification
TCPX11: Unable to decode TYPE-OF-SERVICE attribute in TCP
specification
TCPX12: Unable to decode SECURITY attribute in TCP specification
TCPX13: Unable to decode COMPARTMENTS attribute in TCP specification
TCPX14: unable to decode HANDLING-RESTRICTIONS attribute in TCP
specification
TCPX15: Unable to decode TRANSMISSION-CONTROL attribute in TCP
specification
TCPX16: TCP not initialized and available

Returns a JFN for the specified file. Accepts the specification for the file from both a string in memory and from a file. If both are given as arguments, the string is used first, and then the file is used if more fields are needed to complete the specification. This form also allows the program to specify nonstandard values to be used for omitted fields and to request the assignment of a specific JFN.

ACCEPTS IN AC1: 0 in the left half, and address of the beginning of the argument table in the caller's address space in the right half

AC2: Byte pointer to ASCIZ file specification string in the caller's address space, or 0 if none

RETURNS +1: Failure, error code in AC1
+2: Success, flags in the left half of AC1, and the JFN assigned in the right half of AC1. (This word is called an indexable file handle and is given to the GNJFN call as an argument.) Updated string pointer in AC2, if pertinent.

All I/O errors can occur. These errors cause software interrupts or process terminations, and only a single return (+1) is given.

The format of the argument table specified by the right half of AC1 is described below. Words 0 through 10 (.GJGEN-.GJJFN) must be supplied

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

in the long form of the GTJFN call. The remaining words are optional, and if they are supplied, B15(GJ%XTN) of word .GJGEN must be on.

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .GJGEN | Flag bits in the left half and generation number in the right half. (See below.) |
| 1 | .GJSRC | Input JFN in the left half and output JFN in the right half. To omit either JFN, specify .NULIO (377777). |
| 2 | .GJDEV | Byte pointer to ASCIZ string that specifies the default device to be used when none is given. If this word is 0, the user's connected structure will be used. |
| 3 | .GJDIR | Byte pointer to ASCIZ string that specifies the default directory to be used when none is given. The string should not include brackets around the name. If this word is 0, the user's connected directory will be used. |
| 4 | .GJNAM | Byte pointer to ASCIZ string that specifies the default filename to be used when none is given. If this word is 0, either the string or the input JFN must supply the filename. |
| 5 | .GJEXT | Byte pointer to ASCIZ string that specifies the default file type to be used when none is given. If this word is 0, the null file type will be used. |
| 6 | .GJPRO | Byte pointer to ASCIZ string that specifies the default protection to be used when none is given. If this word is 0, the default protection as specified in the directory or the protection of the next lower generation will be used. |
| 7 | .GJACT | Byte pointer to ASCIZ string that specifies the default account to be used when none is given. If this word is 0, the user's LOGIN account (unless changed) will be used. |
| 10 | .GJJFN | The JFN to associate with the file specification if flag GJ%JFN is set in word 0 (.GJGEN) of the argument block. |

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

11 .GJF2 Extended argument block if B15(GJ%XTN) is on in the left half of .GJGEN. This word contains a second group of flags in the left half and the count of the number of words following this word in the argument block in the right half. The flags in the left half specify additional control over the GTJFN process. The following flags are defined:

B0(G1%RND) Return to the caller if the filename buffer becomes empty, and the user attempts to delete a character. This can occur if the user, when giving the filename, types a CTRL/U or types a DELETE or CTRL/W and there are no more characters in the buffer.

B2(G1%NLN) Filenames cannot be longer than 6 characters and file types cannot be longer than 3 characters. In addition, the generation number, temporary status, protection, and account fields cannot be specified in the string or the input data.

B3(G1%RCM) Return the confirmation message to the caller by placing it in the destination buffer.

B4(G1%RIE) Return to the caller if the input buffer becomes empty, and the user attempts to delete a character.

B5(G1%IIN) Files marked as invisible are to be considered by the system when it is searching for a file to assign to the JFN.

B6(G1%SLN) Prohibit the expansion of logical names. If, for example, user DBELL defines logical name ME: to be PSA:<DBELL> and does a GTJFN for file ME:FOO.BAR, the file specification stored in the JFN block will be:

PSA:<DBELL>FOO.BAR

In this case, the logical name ME: has been expanded to PSA:<DBELL>. However, if bit G1%SLN is set, and a GTJFN performed on file FOO.BAR, the file

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

specification stored in the JFN block is:

ME:FOO.BAR

In this case, the logical name has not been expanded.

B7(G1%LOC) The node name cannot be specified.

- 12 .GJCPP Byte pointer to string where GTJFN is to store the exact copy of the user's typescript (destination string pointer). This string will contain logical names, if they were typed by the user, and will not contain the default fields unless they were generated through recognition. This string allows the caller to obtain a true copy of the user's typescript.
- 13 .GJCPC Number of bytes available in the destination string to which .GTCPP (word 12) points. If a pointer has been specified but this word is 0, the monitor assumes the string contains 130 bytes.
- 14 .GJRTY Byte pointer to the text to be output when the user types a CTRL/R (pointer to the CTRL/R buffer). This pointer cannot be equal to the pointer given in AC2. (See the TEXTI call for the definition of CTRL/R text.)
- 15 .GJBFP Byte pointer to the beginning of the destination buffer. (obsolete)
- 16 .GJATR Pointer to the file specification attribute block.

The attribute block has the following format:

| Word | Contents |
|------|--|
| 0 | Count of words in attribute block (including this word). |
| 1 | Byte pointer to argument string. |
| 1+n | Byte pointer to argument string. |

The ASCIZ argument strings are specified as:

keyword:attribute

The possible keywords and attribute values are as follows:

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

| Keyword | Attribute Value | | | | | | | | | | |
|-------------------------|---|--------|---------|---|----------------------|---|-------------------------|---|---------|---|------------------------------------|
| A: | Installation-defined account string | | | | | | | | | | |
| BDATA: | DECnet binary optional data | | | | | | | | | | |
| BLOCK-LENGTH: | Magnetic-tape block length (in bytes) | | | | | | | | | | |
| BPASSWORD: | DECnet binary password | | | | | | | | | | |
| CHARGE: | DECnet account string | | | | | | | | | | |
| COMPARTMENTS:n | Connection compartmentalization: 16-bit, defaults to 0 (TCP:) | | | | | | | | | | |
| CONNECTION:ACTIVE | | | | | | | | | | | |
| CONNECTION:PASSIVE | Local to foreign connection attribute; defaults to ACTIVE (TCP:) | | | | | | | | | | |
| DATA: | DECnet optional data | | | | | | | | | | |
| EXPIRATION-DATE: | Magnetic-tape expiration date | | | | | | | | | | |
| FOREIGN-HOST:a.b.c.d | Alternative specification for 32-bit foreign host address. "a", "b", "c", and "d" are decimal octets forming the host number. The "." is a required delimiter. A field of zero must be represented as zero. (TCP:) | | | | | | | | | | |
| FORMAT: | Magnetic-tape record format. The argument may be one of the following: | | | | | | | | | | |
| | <table border="0"> <thead> <tr> <th style="text-align: left;">Format</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>F</td> <td>Fixed-length records</td> </tr> <tr> <td>D</td> <td>Variable-length records</td> </tr> <tr> <td>S</td> <td>Spanned</td> </tr> <tr> <td>U</td> <td>Binary files with 36-bits per word</td> </tr> </tbody> </table> | Format | Meaning | F | Fixed-length records | D | Variable-length records | S | Spanned | U | Binary files with 36-bits per word |
| Format | Meaning | | | | | | | | | | |
| F | Fixed-length records | | | | | | | | | | |
| D | Variable-length records | | | | | | | | | | |
| S | Spanned | | | | | | | | | | |
| U | Binary files with 36-bits per word | | | | | | | | | | |
| HANDLING-RESTRICTIONS:n | Connection handling-restrictions option: 16-bit (TCP:) | | | | | | | | | | |
| LOCAL-HOST:a.b.c.d | Alternate specification for 32-bit local host number. See FOREIGN-HOST:a.b.c.d (TCP:) | | | | | | | | | | |
| OFF-LINE | NONE - display-only keyword. The attribute is set by setting bit FB%OFF in word .FBCTL of the FDB block. | | | | | | | | | | |
| P: | Octal file protection value | | | | | | | | | | |

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

PASSWORD: DECnet password string

PERSIST:n

PERSIST:(n,m) Connection opening attempt parameters: 0 to keep trying until successful, n to try for n seconds (default 30), m to try every m seconds (default 5). If no persistence is given, 30 seconds is used. (TCP:)

POSITION: File sequence number to position magnetic-tape to.

RECORD-LENGTH: Magnetic-tape record length (in bytes)

SECURITY:n Connection security field; 16-bit, system default if omitted (TCP:)

T NONE - display-only keyword. The attribute is set by setting bit GJ%TMP in word .GJGEN of the GTJFN block.

TIMEOUT:n Amount of time allowed to pass while waiting for a message from a foreign system. Default is 30 seconds; no timeout if n=0. (TCP:)

TRANSMISSION-CONTROL:n Connection transmission-control option; n is a 24-bit number used by IP (TCP:)

TYPE-OF-SERVICE:n Connection type-of-service indicating tradeoffs made in providing data transmission; n is the low-order 8 bits: default is 0; NET WIZARD, WHEEL or OPERATOR required for other than 0. (TCP:)

USERID: DECnet user ID string

17 .GJNOD Default node

The flag bits accepted in the left half of .GJGEN (word 0) of the argument block are basically the same as those accepted in the short form of the GTJFN call. The entire set of bits is listed below. (See GTJFN - SHORT FORM for more detailed explanations of these bits.) The flags that are different in the two forms are GJ%JFN, GJ%XTN, GJ%FNS, and GJ%SHT.

| Bit | Symbol | Meaning |
|-----|--------|-----------------------------------|
| 0 | GJ%FOU | Create a new version of the file. |

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

| 1 | GJ%NEW | The file must not exist. | | | | | | | | |
|-----------|--|---|-------|---------|-----------|--------------------------|-----------|--|-----------|--|
| 2 | GJ%OLD | The file must exist. | | | | | | | | |
| 3 | GJ%MSG | Type a message if the user presses ESC to terminate input. | | | | | | | | |
| 4 | GJ%CFM | Confirmation from the user is required. | | | | | | | | |
| 5 | GJ%TMP | The file is temporary. | | | | | | | | |
| 6 | GJ%NS | Search only the first specification in a multiple logical name definition. | | | | | | | | |
| 7 | GJ%ACC | The JFN cannot be accessed by inferior processes. | | | | | | | | |
| 8 | GJ%DEL | Ignore the file deleted bit in the FDB. | | | | | | | | |
| 9-10 | GJ%JFN | Associate the JFN supplied in .GJJFN (word 10) of the argument block with the file specification. The value of this field is interpreted as follows: | | | | | | | | |
| | | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>0(.GJDNU)</td> <td>Ignore the JFN supplied.</td> </tr> <tr> <td>2(.GJERR)</td> <td>Attempt to assign the JFN supplied and return an error if it is not available.</td> </tr> <tr> <td>3(.GJALT)</td> <td>Attempt to assign the JFN supplied and, if it is not available, assign an alternate.</td> </tr> </tbody> </table> | Value | Meaning | 0(.GJDNU) | Ignore the JFN supplied. | 2(.GJERR) | Attempt to assign the JFN supplied and return an error if it is not available. | 3(.GJALT) | Attempt to assign the JFN supplied and, if it is not available, assign an alternate. |
| Value | Meaning | | | | | | | | | |
| 0(.GJDNU) | Ignore the JFN supplied. | | | | | | | | | |
| 2(.GJERR) | Attempt to assign the JFN supplied and return an error if it is not available. | | | | | | | | | |
| 3(.GJALT) | Attempt to assign the JFN supplied and, if it is not available, assign an alternate. | | | | | | | | | |
| 11 | GJ%IFG | The file specification can contain wildcard characters. | | | | | | | | |
| 12 | GJ%OFG | Associate the JFN with the file specification string and not the file itself. This is termed a "parse-only JFN", and allows the syntax of a file name to be checked regardless of whether or not a file of that name actually exists. | | | | | | | | |
| 13 | GJ%FLG | Return flags in AC1 on successful completion of the call. | | | | | | | | |
| 14 | GJ%PHY | The physical device is to be used. | | | | | | | | |
| 15 | GJ%XTN | The argument block contains more than 10 (octal) words. This bit must be set for the long form. | | | | | | | | |
| 16 | GJ%FNS | This bit is ignored for the long form of the GTJFN call. | | | | | | | | |

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

17 GJ%SHT This bit must be off for the long form of the GTJFN call.

The generation number given in the right half of .GJGEN (word 0) of the argument block can be one of the following:

- 0(.GJDEF) to indicate that the next higher generation number is to be used if GJ%FOU is on, or to indicate that the highest existing generation number is to be used if GJ%FOU is off.
- 1(.GJNHG) to indicate that the next higher generation number is to be used if no generation number is supplied.
- 2(.GJLEG) to indicate that the lowest existing generation number is to be used if no generation number is supplied.
- 3(.GJALL) to indicate that all generation numbers are to be used and that the JFN is to be assigned to the first file in the group, if no generation number is supplied. (Bit GJ%IFG must be on.)
- 1-377777 to indicate that the specified number is to be used as the generation if no generation number is supplied.

On a successful return, the following flags are returned in the left half of AC1 if flag bit GJ%IFG, GJ%OFG, or GJ%FLG was on in the call.

Bits Returned on Successful GTJFN Call

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | GJ%DEV | The device field of the file specification contained wildcard characters. |
| 1 | GJ%UNT | The unit field of the file specification contained wildcard characters. This bit will never be set because wildcard characters are not allowed in unit fields. |
| 2 | GJ%DIR | The directory field of the file specification contained wildcard characters. |
| 3 | GJ%NAM | The filename field of the file specification contained wildcard characters. |
| 4 | GJ%EXT | The file type field of the file specification contained wildcard characters. |
| 5 | GJ%VER | The generation number field of the file specification contained wildcard characters. |

TOPS-20 MONITOR CALLS
(GTJFN Long Form)

| | | |
|----|--------|--|
| 6 | GJ%UHV | The file used has the highest generation number because a generation number of 0 was given in the call. |
| 7 | GJ%NHV | The file used has the next higher generation number because a generation number of 0 or -1 was given in the call. |
| 8 | GJ%ULV | The file used has the lowest generation number because a generation number of -2 was given in the call. |
| 9 | GJ%PRO | Protection field of file specification given |
| 10 | GJ%ACT | The account field of the file specification was given. |
| 11 | GJ%TFS | The file specification is for a temporary file. |
| 12 | GJ%GND | Files marked for deletion were not considered when assigning JFNs. This bit is set if GJ%DEL was not set in the call. |
| 13 | GJ%NOD | The node name field of the file specification was given. |
| 17 | GJ%GIV | Invisible files were not considered when assigning JFNs. This bit is set by the monitor if G1%IIN was not set by the user in the GTJFN call. |

See the short form of the GTJFN call for the possible error mnemonics.

Returns the paging trap information for the specified process.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with
AC1 Containing number of pager traps (the number of times a trap has occurred to the pager) for designated process since the process was started

TOPS-20 MONITOR CALLS
(GTRPI)

- AC2 Containing number of page faults (the number of times a trap has resulted in a page being swapped in) for designated process since the process was started
- AC3 Containing time spent (in milliseconds) in page routines by designated process since the process was started

The number of pager traps will be greater than or equal to the number of page faults.

Generates an illegal instruction interrupt on error conditions below.

GTRPI ERROR MNEMONICS:

- FRKHX1: Invalid process handle
- FRKHX2: Illegal to manipulate a superior process
- FRKHX3: Invalid use of multiple process handle

Returns the trap words. This monitor call allows a program to retrieve information about a previous read, write, or execute trap.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with trap status word from last memory trap in AC1, and last monitor call that had an error in AC2.

The following bits are defined in the status word:

- B0(PF%USR) page failure-user mode reference
- B5(PF%WRT) page failure-write reference
- B14(TSW%RD) trap status-read (always on)
- B15(TSW%WT) trap status-write (same setting as B5)
- B16(TSW%EX) trap status-execute (always on)
- B17(TSW%MN) trap status-monitor mode reference (complement of B0)
- B18-35 address of reference that caused the trap

This information allows a program to determine the exact cause of a memory trap and/or the effective virtual address that caused the trap. This information is sufficient to enable the program to continue, if desired, when the cause of the trap has been removed.

The contents of AC1 is 0 if there have been no memory traps.

TOPS-20 MONITOR CALLS
(GTRPW)

Generates an illegal instruction interrupt on error conditions below.

GTRPW ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

Returns the status of a file associated with a JFN.

ACCEPTS IN AC1: JFN in the right half

RETURNS +1: Always, with status in AC2. If JFN is illegal in any way, B10 of AC2 will be 0.

JFN STATUS WORD

B0(GS%OPN) file is open
B1(GS%RDF) if file is open (if bit 0 is on), it is open for read access
B2(GS%WRF) if file is open, it is open for write access
B3(GS%XCF) if file is open, it is open for execute access
B4(GS%RND) if file is open, it is open for non-append access
B7(GS%LNG) file is longer than 512 pages
B8(GS%EOF) last read was past end of file
B9(GS%ERR) file may be in error (a device or data error occurred)
B10(GS%NAM) file specification is associated with this JFN
B11(GS%AST) the JFN is parse-only (GJ%OFG was set in GTJFN call)
B12(GS%ASG) JFN is currently being assigned
B13(GS%HLT) I/O errors are considered terminating conditions
B17 This is a restricted JFN (GJ%ACC was set in GJTJFN call). Only the process that received this JFN may use it. Other processes may get another JFN for this file.
B18(GS%PLN) if set, any line numbers present in the file are passed to the program during input (SIN, BIN, etc). If zero, line numbers are stripped from the data passed to the program.
B32-35 data mode of the file. See Chapter 2.
(GS%MOD)

TOPS-20 MONITOR CALLS
(GTSTS)

| | | |
|----|--------|-------------------|
| 0 | .GSNRM | normal data mode |
| 1 | .GSSMB | small buffer mode |
| 10 | .GSIMG | image mode |
| 17 | .GSDMP | dump mode |

If B0(GS%OPN) is not set on return, the file is not opened, and the settings of bits 1 through 4 are indeterminate.

The STSTS call can be used to set the status of a particular file.

Returns the terminal type number for the specified terminal line.
(See Section 2.4.9.4 for the terminal type numbers.)

ACCEPTS IN AC1: Terminal designator

RETURNS +1: Always, with terminal type number in AC2 and buffer allocation numbers (# of input buffers to be allocated in left half, and # of output buffers to be allocated in right half) in AC3. AC1 is unchanged.

The STTYP monitor call can be used to set the terminal type number for a specified line.

Generates an illegal instruction interrupt on error conditions below.

GTTY ERROR MNEMONICS:

DESX1: Invalid source/destination designator
TTYX01: Line is not active

Halts the current process and any inferior processes of the current process. Sets the process's PC to the next after the call and saves it in the Process Storage Block (PSB) in case the process is continued. The user can continue the process by typing the CONTINUE command, which causes the process to start at the next instruction.

TOPS-20 MONITOR CALLS
(HALTF)

Sets bits 1-17(RF%STS) in the status word for this process to 2(.RFVPT). See the RFSTS monitor call for the format of the status word.

If the top level process executes a HALTF call and does not have WHEEL or OPERATOR capability enabled, the job is logged out. If the top level process executes a HALTF call and does have WHEEL or OPERATOR capability enabled, control passes to mini-exec level.

Halts one or more inferior processes. (See the HALTF monitor call description to halt the current process.)

ACCEPTS IN AC1: Process handle (inferior processes only)

RETURNS +1: Always

Sets bits 1-17(RF%STS) in the status word(s) for addressed process(s) to 2(.RFVPT). See the RFSTS monitor call for the format of the status word.

Generates an illegal instruction interrupt on error conditions below.

HFORK ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
HFRHX1: Illegal to halt self with HFORK

Returns the value of one of the high precision system clocks. Although the main time base from interrupts generated by the internal system clock is in units of 1 millisecond, the clock provides a time base in units of 10 microseconds. The HPTIM monitor call provides access to the variables kept in these high precision units.

ACCEPTS IN AC1: Number of the clock to read (see below)

TOPS-20 MONITOR CALLS
(HPTIM)

RETURNS +1: Failure, error code in AC1
 +2: Success, with AC1 containing the value of the
 specified clock

The numbers for currently-defined clocks are:

| | | |
|---|--------|--|
| 0 | .HPELP | Elapsed time since system startup. (See the TIME call for obtaining the time in milliseconds.) |
| 1 | .HPRNT | CPU runtime for this process. (See the RUNTM call for obtaining the time in milliseconds.) |

HPTIM ERROR MNEMONICS:

HPTX1: Undefined clock number

Initiates an orderly shutdown of the timesharing operation of the system. This call causes periodic notices of the impending shutdown to be issued to all terminals. It also causes any jobs still logged in at the designated shutdown to be logged out.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Shutdown time with the date and time in the internal format. (See Section 2.9.2.)

AC2: Date and time in internal format when system operation will resume (or 0 if unknown). Used for advisory messages only.

RETURNS +1: Failure, error code in AC1
 +2: Success, shutdown procedure initiated

The shutdown notice is issued immediately to all terminals if the shutdown time is within two hours. The notice is also sent two hours, one hour, 30 minutes, 10 minutes, 5 minutes, and one minute before the shutdown.

The time when the system is expected to be placed back into operation is not used directly by the monitor. It is entered into a GETAB table where it may be examined with the GETAB monitor call.

TOPS-20 MONITOR CALLS
(HSYS)

HSYS ERROR MNEMONICS:

CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required
TIMEX1: Time cannot be greater than 24 hours
TIMEX2: Downtime cannot be more than 7 days in the future

Converts separate numbers for the local year, month, day, and time into the internal date and time format. (See Section 2.9.2 for more information on the internal format.)

ACCEPTS IN AC2: Year in the left half, and numerical month (0=January) in the right half

AC3: Day of the month (0=first day) in the left half, and 0 in the right half

AC4: B0(IC%DSA) Apply daylight savings according to the setting of B1(IC%ADS). If B0 is off, daylight savings is applied only if appropriate for the date.

B1(IC%ADS) Apply daylight savings if B0(IC%DSA) is on.

B2(IC%UTZ) Use time zone in B12-17. If this bit is off, the local time zone is used.

B3(IC%JUD) Interpret the number in the right half of AC2 as being in Julian day format (Jan 1 is day 1).

B12-17 Time zone to use if B2(IC%UTZ) is on.
(IC%TMZ) (See Section 2.9.2 for the time zones.)

B18-35 Local time in seconds since midnight.
(IC%TIM)

RETURNS +1: Failure, error code in AC1
+2: Success, AC2 contains the internal date and time, and AC3 contains
B0 and B2 On for compatibility with the ODCNV call

TOPS-20 MONITOR CALLS
(IDCNV)

B1(IC%ADS) On if daylight savings was applied
B12-17 Time zone used
(IC%TMZ)

IDCNV ERROR MNEMONICS:

DATEX1: Year out of range
DATEX2: Month is not less than 12
DATEX3: Day of month too large
DATEX5: Date out of range
DATEX7: Julian day is out of range
TIMEX1: Time cannot be greater than 24 hours
ZONEX1: Time zone out of range

Inputs the date and time and converts them to the internal date and time format. (See Section 2.9.2.) The IDTIM monitor call does not permit either the date or the time to be entered separately and does not perform conversions for time zones other than the local one (unless the time zone is specified in the input string). See the IDTNC and IDCNV monitor calls descriptions for these functions.

ACCEPTS IN AC1: Source designator

AC2: Format option flags (see below), 0 is the normal case

RETURNS +1: Failure, error code in AC2, updated string pointer in AC1, if pertinent

+2: Success, updated string pointer, if pertinent, in AC1, and the internal format date and time in AC2

The format option flags in AC2 specify the interpretation to be used when a date or time specification is ambiguous.

IDTIM Option Flags

B1(IT%NNM) Do not allow the month to be numeric and ignore B2-3.

B2(IT%SNM) Interpret the second number in the date as the month (for example, 6/2/76 is interpreted as Feb. 6, 1976).

TOPS-20 MONITOR CALLS
(IDTIM)

If this bit is off, the first number is interpreted as the month (for example, 2/6/76 is interpreted as Feb. 6, 1976).

B3(IT%ERR) Return an error if the order of the day and month does not agree with the setting of B2(IT%SNM) even though the date can be successfully interpreted. If this bit is off, a date which can be interpreted by assuming the day and month are in the opposite order than that specified by the setting of B2(IT%SNM) will be considered valid. For example, if B2-3 are off, 30/5/76 will be considered as a valid date.

B7(IT%NIS) Seconds cannot be included in a time specification.

B8(IT%AIS) Seconds must be included in a time specification and must be preceded by a colon.

If B7-8 are both off, seconds are optional in a time specification. If specified, seconds must be preceded by a colon.

B9(IT%NAC) Colon cannot be used to separate hours and minutes.

B10(IT%AAC) Colon must be used to separate hours and minutes.

If B9-10 are both off, a colon is optional between hours and minutes.

B11(IT%AMS) When B7-10 are off, always interpret a time specification containing one colon as hhmm:ss.

B12(IT%AHM) When B7-10 are off, always interpret a time specification containing one colon as hh:mm and return an error if the first field is too large. This differs from B7(IT%NIS) in that seconds can be included if preceded by a second colon.

If B7-12 are all off, a time specification containing one colon is interpreted as hh:mm if the first field is small enough. Otherwise it is interpreted as hhmm:ss.

B14(IT%N24) Do not allow the time to be specified in 24-hour format (for example, 1520 for 3:20 in the afternoon) and make AM or PM specification mandatory.

B15(IT%NTM) Do not allow the time specification to include AM, PM, NOON, or MIDNIGHT.

B16(IT%NTZ) Do not allow a time zone to be specified.

TOPS-20 MONITOR CALLS
(IDTIM)

If AC2 is 0, the IDTIM call accepts the date and time in month/day/year or day/month/year format. Hyphens (-), slashes (/), and spaces () are valid delimiters. In cases where pure numeric representation is used for the date (1/9/1967, for example), IDTIM checks the first number for being in the range: $0 < n < 13$. If the test is successful, the first number is interpreted as the month. If the test is unsuccessful, the test is made on the second number and if successful, that number is interpreted as the month. Otherwise an error is generated. For example:

1. 5/6/1976 is interpreted as May 6, 1976
2. 6/5/1976 is interpreted as June 5, 1976
3. 13/5/1976 is interpreted as May 13, 1976
4. 13/13/1976 generates an error

IDTIM ERROR MNEMONICS:

DILFX1: Invalid date format
TILFX1: Invalid time format
DATEX1: Year out of range
DATEX3: Day of month too large
DATEX5: Date out of range

All I/O errors are also possible. These errors cause software interrupts or process terminations as described under the BIN call.

Inputs the date and/or the time and converts it into separate numbers for the local year, month, day, or time. The IDTNC call allows the date or time to be entered separately, which is not possible with the IDTIM JSYS because neither one can be converted to the internal format without converting the other. (See Section 2.9.2.)

ACCEPTS IN AC1: Source designator

AC2: Format option flags
In addition to the flags described in the IDTIM call, the flags below can also be specified:

B0(IT%NDA) Do not input the date and ignore B1-3. If IT%NDA is off, the date must be input.

TOPS-20 MONITOR CALLS
(IDTNC)

B6(IT%NTI) Do not input the time and ignore B7-16.
If IT%NTI is off, the time must be input.

RETURNS +1: Failure, error code in AC2, updated string pointer,
if pertinent, in AC1

+2: Success, updated string pointer, if pertinent, in AC1

If the date was input,

AC2 contains the year in the left half, and the month
(0=January) in the right half.

AC3 contains the day of the month (0=first day) in
the left half, and the day of the week (0=Monday)
in the right half.

If the time was input,
AC4 contains

B0(IC%DSA) On if IT%NTI was set in AC2, or if
IT%NDA was set in AC2 and a time zone
was input (for compatibility with the
ODCNV call).

B1(IC%ADS) On if a daylight savings time zone
was input, or if IT%NTI was set in
AC2.

B2(IC%UTZ) On if IT%NTI was set in AC2, or if
IT%NDA was set in AC2 and a time zone
was input (for compatibility with the
ODCNV call).

B3(IC%JUD) On if a number in Julian day format
was input.

B12-17 The time zone if one was input, or
(IC%TMZ) The local time zone if none was
input. (See Section 2.9.2 for the
time zones.)

B18-35 Time as seconds since midnight.
(IC%TIM)

A -1 returned in both AC2 and AC3 means the system date and time have
not been set.

IDTNC ERROR MNEMONICS:

DILFX1: Invalid date format

TILFX1: Invalid time format

All I/O errors are also possible. These errors cause software
interrupts or process terminations as described under the BIN call
description.

TOPS-20 MONITOR CALLS
(IDTNC)

The IDTNC call does not detect certain errors in date input, such as day 31 of a 30-day month. These errors are detected by the IDCNV call.

Initiates software interrupts on the specified channels in a process. (See Section 2.6.)

ACCEPTS IN AC1: Process handle

AC2: 36-bit word
Bit n on means initiate a software interrupt on channel n.

RETURNS +1: Always

Generates an illegal instruction interrupt on error conditions below.

IIC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Allows the user to obtain specific information on either the current system or any other system within a cluster.

RESTRICTIONS: Requires WHEEL or OPERATOR capabilities enabled.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Always, with

AC1: Address of argument block; or
1B0 - IN%RER to indicate a remote error, in which case, right half is remote error code.

TOPS-20 MONITOR CALLS
(INFO%)

CAUTION

Upon successful return from INFO%, ACs 1-4 are not modified. This is due to the fact that INFO% preserves these ACs because it may perform internal monitor calls with the user's AC block. It is suggested that you do not attempt to have information returned by INFO% in ACs 1-4.

The format of the argument block is as follows:

NOTE

The length of the argument block includes the .INFUN word. Argument block must be at least .INMIN words (.INMIN=3) long and no more than .INMAX(6).

| | Left Half | Right Half |
|--------|--|----------------------|
| .INFUN | Function code | Length of arg. block |
| .INCID | CI node number (-1 for the local system) | |
| .INAC1 | INFO% Functions | |
| .INAC2 | INFO% Functions | |
| .INAC3 | INFO% Functions | |
| .INAC4 | See individual functions | |

Code Symbol Meaning

0 .INCIN This function returns a maximum of 16 36-bit words. The CI node numbers of the systems responding to requests for remote information are returned in each word.

Argument Block:

.INAC1 Address of block to return CI node numbers.

.INCIN Returns an argument block as the following diagram illustrates.

TOPS-20 MONITOR CALLS
(INFO%)

| | |
|---|--|
| 0 | Length of block returned |
| 1 | CI node number of first (local) system |
| 2 | CI node number of second system |
| 3 | CI node number of third system |
| n | CI node number of last system |

- 1 .INCFG This function causes the CNFIG% monitor call to be executed on the specified system. See the CNFIG% monitor call for more information.

Argument block:

.INAC1 This word contains a function code for the CNFIG% on the specified system. (See CNFIG% - AC1)

.INAC2 This contains the address of the argument block for CNFIG%. (See CNFIG% - AC2)

- 2 .INDST This function causes a DIRST% monitor call to be performed on the specified system. (See DIRST% for more information).

Argument block:

.INAC1 Destination designator

.INAC2 User or directory number

- 3 .INGTB This function allows a GETAB% monitor call to be performed on the specified system. See GETAB% for more information. This function returns the 36-bit word from the table specified in .INAC1 in .INAC2 on success.

Argument block:

.INAC1 Index into table in left half, and table number in right half. (See Section 2.3.2 in the Monitor Calls Reference Manual.) See GETAB% for more information.

- 4 .INGJI This function performs a GETJI% monitor call.

TOPS-20 MONITOR CALLS
(INFO%)

Argument block:

- .INAC1 Job number or .TTDES+TTY number (-1 does not apply to this function)
 - .INAC2 -<length of destination block>,,address of block. See GETJI% for a description of the block.
 - .INAC3 Offset of first entry desired from job information table
- 5 .INGTY This function works like the GTTYP% monitor call and returns the information in the same manner that GTTYP% does. See the GTTYP% monitor call for more information.

Argument Block:

- .INAC1 Terminal designator
- 6 .ININL This function does a INLNM% using only the .INSLY function.

Argument Block:

- .INAC1 0 in the left half, and index into the table of logical names in the right half. (See AC1 for INLNM%.)
 - .INAC2 Byte pointer to the string for storing the logical name. (See AC2 for INLNM%.)
- 7 .INLNS This function enables a LNMST% to be performed using only the .LNSSY function.

Argument Block:

- .INAC1 .LNSSY
 - .INAC2 Pointer to the logical name. The logical name must contain a colon. (See AC2 for LNMST%.)
 - .INAC3 Pointer to the string where the original logical name definition is to be written. The name returned includes a terminating colon. (See AC3 for LNMST%.)
- 10 .INMSR Performs MSTR% functions as listed. Some functions require WHEEL or OPERATOR capability enabled.

TOPS-20 MONITOR CALLS
(INFO%)

Argument Block:

.INAC1 Length of argument block in the left half and
 function code in right half (see below).

.INAC2 Address of argument block (see MSTR% for
 format).

Only the following MSTR% functions are valid for .INMSR
(see MSTR% for more information):

| Function | Symbol | Privileged | Meaning |
|----------|--------|------------|---|
| 0 | .MSRNU | Yes | Return status of next disk unit |
| 1 | .MSRUS | Yes | Return status of given disk unit |
| 4 | .MSGSS | No | Return status of given structure |
| 11 | .MSGSU | No | Return the job numbers of the users on the given structure |

11 .INMTO Performs MTOPR% functions as listed below.

Argument Block:

.INAC1 TTY device designator

.INAC2 Function (see below)

.INAC3 Address of argument block (if necessary)

Only the following MTOPR% functions are available (see
MTOPR% for more information):

| | | | |
|--------|--------|--------|--------|
| .MOPIH | .MORSP | .MORLW | .MORLL |
| .MORNT | .MORBM | .MORFW | .MORXO |
| .MORLC | .MORLM | .MOPCR | .MORTF |
| .MORTC | .MOCTM | | |

12 .INMUT Performs a MUTIL% monitor call on the given system.
See MUTIL% for more information.

Argument Block:

.INAC1 Length of argument block

.INAC2 Address of argument block

TOPS-20 MONITOR CALLS
(INFO%)

Only the following functions of the MUTIL% monitor call can be executed:

.MUGTI .MUFOJ .MUFSQ .MUFFP
.MUFPQ .MURSP .MUMPS

- 13 .INRCR Performs an RCUSR% on the specified system. This function returns the same information as RCUSR%. See RCUSR% for more information.

Argument Block:

.INAC1 Flag bits in the left half
.INAC2 Byte pointer of ASCII string to be translated
.INAC3 36-bit user number (given when stepping to the next user name in a group)

- 14 .INSKD Performs a SKED% on the specified system. See SKED% for more information.

Argument Block:

.INAC1 Function Code
.INAC2 Address of argument block

Only the following functions can be done. See SKED% for more information about them:

.SKRBC .SKRCS .SKRJP
.SKBCR .SKRCV

- 15 .INSNP Performs only 2 SNOOP% functions on the specified system. These functions are .SNPSY and .SNPAD. See SNOOP% for more information. Requires WHEEL, OPERATOR or MAINTENANCE capability enabled.

Argument Block:

.INAC1 Function code (.SNPSY or .SNPAD)
.INAC2 Function-specific argument
.INAC3 Function-specific argument

- 16 .INSGT Returns the table number, table length, and word 0 of the specified system table for the specified system. (See Section 2.3.2 of the Monitor Calls Reference Manual for the names of the system tables.) See SYSGT% for more information.

TOPS-20 MONITOR CALLS
(INFO%)

Argument Block:

.INAC1 SIXBIT table name

- 17 .INTMN Performs a TMON%. See the TMON% monitor call for more information.

Argument Block:

.INAC1 Function code (see TMON%)

- 20 .INXPK Performs an XPEEK%. This function requires WHEEL or OPERATOR capability enabled. See XPEEK% for more information. Note that this function cannot return more than one page (512 36-bit words) of data.

Argument Block:

.INAC1 Address of argument block

- 21 .INDVC Performs a DVCHR% monitor call. See the DVCHR% monitor call for more information.

Argument Block:

.INAC1 Device designator

- 22 .INNTF Performs a NTINF% monitor call on the specified system. See NTINF% for more information.

Argument Block:

.INAC1 Address of argument block. Note that word .NWLIN of the argument block cannot contain -1.

- 23 .INSTV Performs a STDEV% monitor call. See STDEV% for more information.

Argument Block:

.INAC1 Byte pointer to the string to be translated.

- 24 .INDVT Performs a DEVST% monitor call. See DEVST% for more information.

Argument Block:

.INAC1 Destination designator

.INAC2 Device designator

TOPS-20 MONITOR CALLS
(INFO%)

25 .INSYS Returns SYSTAT string information. The argument block held in .INAC1 contains the byte pointers where the monitor is to return the information.

Argument Block:

.INAC1 Address of argument block to return information (see format of argument block below)

.INAC2 Job number or .TTDES+TTY number

| | | |
|----|--------|---|
| 0 | .SYUSR | Byte pointer to store username |
| 1 | .SYDIR | Byte pointer to store connected directory |
| 2 | .SYPRG | SIXBIT program name |
| 3 | .SYORG | Byte pointer to job origin |
| 4 | .SYCJB | Controlling job number |
| 5 | .SYTTY | Controlling terminal number |
| 6 | .SYJOB | Job number |
| 7 | .SYSTT | 0 if state is TI, 1 if state is RUN |
| 10 | .SYTIM | Job runtime |
| 11 | .SYLIM | Job runtime limit |
| 12 | .SYCLS | Job Class (class scheduling) |
| 13 | .SYSHR | Job Share (class scheduling) |
| 14 | .SYUSE | Job Use (class scheduling) |
| 15 | .SYJCT | Job's connect time |

26 .INJOB This function returns a block of data containing the job numbers and terminal numbers for the given user.

Argument Block:

.INAC1 Byte pointer to username

.INAC2 Address of argument block (see below)

TOPS-20 MONITOR CALLS
(INFO%)

This function returns information in the argument block specified in .INAC2 as follows:

| | | | | |
|---------|--------|--|------------------------------|-----------------|
| 0 | .JOLEN | | Count of words in this block | |
| | | | | |
| 1 | | | Job number | Terminal number |
| | | | | |
| 2 | | | Job number | Terminal number |
| | | | | |
| /-----/ | | | | |
| / | | | | |
| / | | | | |
| | | | | |
| n | | | Job number | Terminal number |
| | | | | |

This function returns a slot in the argument block for each job that the specified user is logged into on the requested system. The count specified in the .JOLEN word includes the .JOLEN word. If the user is not logged into the specified node, this function returns an INFX07 error.

- 27 .INRCD Performs an RCDIR% JSYS call on the specified system. This function returns the same information as the RCDIR% monitor call. (See RCDIR% for more information.)

Argument Block:

- .INAC1 Flag bits in the left half.
- .INAC2 Byte pointer of ASCII string to be translated.
- .INAC3 36-bit directory number (given when stepping to the next user name in a group).

- 30 .INTIM Performs a TIME% JSYS call on the specified system. This function returns the same information as the TIME% monitor call. (See TIME% for more information.)

Argument Block:

- .INAC1 System uptime in milliseconds (returned).

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(INFO%)

INFO% ERROR MNEMONICS

INFX01: Invalid INFO% function
INFX02: Invalid CI node number
INFX03: WHEEL or OPERATOR capability required
INFX04: CI node disconnected before information was returned
INFX05: Remote node not supplying information
INFX06: Insufficient system resources - no more swappable free space
INFX07: User not logged in
INFX08: Insufficient system resources on remote system
INFX09: Unimplemented function on remote system
INFX10: Insufficient SCA buffers to process request
INFX11: Remote system not running CLUDGR SYSAP
INFX12: Invalid argument block
INFX13: Job not logged in
INFX14: Remote node could not execute given function
INFX15: Bad argument block length
INFX16: Insufficient credit to send request to remote system
INFX17: Remote XPEEK% can only return 512 words

All I/O errors can occur also.

Returns a logical name that is defined either for this job or for the system. (See Section 2.2.2 and CRLNM and LNMST monitor calls.)

ACCEPTS IN AC1: Function code in the left half, and index into the table of defined logical names in the right half

AC2: Byte pointer to the string for storing the logical name

RETURNS +1: Failure, error code in AC1

+2: Success, updated string pointer in AC2

The available functions are:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .INLJB | List the logical names defined for this job |
| 1 | .INLSY | List the logical names defined for the system |

TOPS-20 MONITOR CALLS
(INLNM)

INLNM ERROR MNEMONICS:

INLNX1: Index is beyond end of logical name table
INLNX2: Invalid function

Performs Internet protocol network management operations.

RESTRICTIONS: Requires NET WIZARD capability enabled.

ACCEPTS IN AC1: Function code

AC2: Function dependent argument

AC3: Function dependent argument

RETURNS +1: Always, with error code in AC1 on failure

Function Codes:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .IPSNT | Change network state. AC2 contains the Internet network number and AC3 contains the desired network state (zero to disable; nonzero to enable). |
| 1 | .IPRNT | Read network state. AC2 contains the Internet network number. The network state is returned in AC3 (zero for disabled; nonzero for enabled). |
| 2 | .IPINI | Reload Internet host and nameserver tables. |
| 3 | .IPGWY | Reload Internet gateway routing table. |
| 4 | .IPRIB | Read status of internet bypass. |
| 5 | .IPSIB | Set status of internet bypass. |
| 6 | .IPNIP | Enable/Disable NI IP protocol operations. |
| 7 | .IPNAP | Enable/Disable NI ARPANET protocol operations. |
| 10 | .IPIGH | Reload NI Internet Protocol. |

TOPS-20 MONITOR CALLS
(IPOPR%)

- 11 .IPRGH Return NI Internet Protocol GHT table.
- 12 .IPRIC Return NI Internet Protocol portal counters.
- 13 .IPRAC Return NI ARP protocol portal counters.
- 14 .IPDNS Reload Internet nameserver table.

IPOPR% ERROR MNEMONICS:

- TCPX23: Invalid IPOPR function requested
- TCPX24: Wheel, Operator, or Network Wizard needed for special IPOPR function
- IPHCHK: Computed GHT checksum does not match
- IPHCNT: GHT entry count argument is not correct
- IPHNSP: Insufficient system resources (No free space for GHT)
- IPHEMX: Exceeded maximum number of GHT entries
- IPHSEQ: GHT Internet host numbers not in ascending order
- IPFLAD: Local Internet host number not in GHT
- ARPNSP: Insufficient system resources (No space for ARP buffers)
- IPARPl: Cannot start ARP until TCPNI service is running
- TCPX44: Monitor does not support TCP over Ethernet

Returns the file specification currently associated with the JFN.

ACCEPTS IN AC1: Destination designator where the ASCIIZ string is to be written

AC2: Indexable file handle (see GTJFN), or pointer to string

AC3: Format control bits to be used when returning the string, or 0

AC4: Byte pointer to string containing prefix of file specification attribute

RETURNS +1: Always, with updated string pointer, if pertinent, in AC1

AC2 can have one of two formats, depending on B26(JS%PTR) in AC3. The first format is a word with either 0 or the flag bits returned from GTJFN in the left half and the JFN in the right half. When the left

TOPS-20 MONITOR CALLS
(JFNS)

half is 0, the string returned is the exact specification associated with the JFN. If the given JFN is associated only with a file specification (it was obtained with B12(GJ%OFG) on in the GTJFN call), the string returned contains null fields for nonexistent fields or fields containing wildcards, and actual values for existent fields.

When the left half is nonzero, the string returned contains wildcard characters for appropriate fields and 0, -1, or -2 as a generation number if the corresponding bit is on in the call.

The second format (allowed only if B26(JS%PTR) of AC3 is on) is a pointer to the string to be returned. This string is one field of a file specification. The field is determined by the first nonzero 3-bit field in AC3 or by the setting of B27(JS%ATR) or B28(JS%AT1) in AC3. For example, if bits 6-8 (JS%NAM) of AC3 are nonzero, then the string is interpreted as a filename field. If B27(JS%ATR) is on, the string is interpreted as a file specification attribute. If B28(JS%AT1) is on, the string is concatenated to the string to which AC4 points, and a colon is inserted between the two strings. In all cases, the string is output to the destination designator, and the appropriate punctuation is added.

AC3 contains control bits for formatting the string being returned. B0-20 are divided into fields corresponding to the fields in a file specification. The value of the control bits determines the output for that field of the file specification. The values are:

- 0 (.JSNOF) do not output this field
- 1 (.JSAOF) always output this field
- 2 (.JSSSD) suppress this field if it is the system default

The bits that can be set in AC3 are as follows:

- B0(JS%NOD) Output for node field
- B1-2(JS%DEV) Output for device field
- B3-5(JS%DIR) Output for directory field
- B6-8(JS%NAM) Output for filename field (2 is illegal)
- B9-11(JS%TYP) Output for file type field (2 is illegal)
- B12-14(JS%GEN) Output for generation number field
- B0-14(JS%SPC) Output for all file specification fields named above. This field should have the same bits set as would be set in the fields above. (See B35(JS%PAF) below.)
- B15-17(JS%PRO) Output for protection field
- B18-20(JS%ACT) Output for account field
- B21(JS%TMP) Return ;T if appropriate
- B22(JS%SIZ) Return size of file in pages
- B23(JS%CDR) Return creation date
- B24(JS%LWR) Return date of last write
- B25(JS%LRD) Return date of last read
- B26(JS%PTR) AC2 contains pointer to the string being returned

TOPS-20 MONITOR CALLS
(JFNS)

| | |
|-------------|--|
| B27(JS%ATR) | Return file specification attributes if appropriate |
| B28(JS%AT1) | Return the specific specification attribute whose prefix is indicated by the string to which AC4 points. This bit is used when a program is processing attributes one at a time. If JS%ATR is also set, all attributes will be returned (WHEEL capabilities are required to receive the password). See the description of the long-form GTJFN for a list of file attributes. |
| B29(JS%OFL) | Return the "OFFLINE" attribute |
| B32(JS%PSD) | Punctuate the size and date fields |
| B33(JS%TBR) | Tab before all fields returned, except for first field |
| B34(JS%TBP) | Tab before all fields that may be returned (fields whose value is given as 1 or 2), except for first field |
| B35(JS%PAF) | Punctuate all fields from node through ;T |

If B32-35 are 0, punctuation between fields is not used.

If AC3 is 0, the string is output in the format

```
node::dev:<directory>name.typ.gen;T
```

The temporary attribute (;T) is not returned if the JFN is a parse-only JFN (see GJ%OFG in the GTJFN description) or the file is not temporary.

The punctuation used on each field is shown below.

```
dev:<directory>name.typ.gen;attribute  
,size,creation date,write date,read date
```

The GTJFN or GNJFN monitor call is used to associate a JFN with a given file specification string.

Generates an illegal instruction interrupt on error conditions below.

JFNS ERROR MNEMONICS:

| | |
|--------|--|
| DESX1: | Invalid source/destination designator |
| DESX2: | Terminal is not available to this job |
| DESX3: | JFN is not assigned |
| DESX4: | Invalid use of terminal designator or string pointer |
| IOX11: | Quota exceeded |
| IOX34: | Disk full |
| IOX35: | Unable to allocate disk - structure damaged |

TOPS-20 MONITOR CALLS
(KFORK)

Kills one or more processes. When a process is killed, all private memory acquired by the process and its Process Storage Block are released. Also, any JFNs the process has created are released, and any terminal interrupt assignments that were acquired from another process are passed back. (Note that because the process is deleted asynchronously, a page of a file mapped into a lower process may not be unmapped before the KFORK call returns.)

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, unless the current process attempts to kill itself

The KFORK call will not release a process handle that identifies a process already killed by another process. In this case, the RFRKH call must be used to release the handle.

The CFORK monitor call can be used to create an inferior process.

Generates an illegal instruction interrupt on error conditions below.

KFORK ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle
KFRK1: Illegal to kill top level process
KFRK2: Illegal to kill self

Performs Local Area Transport (LAT) functions for TOPS-20.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Always

The possible LATOP% functions are as follows:

TOPS-20 MONITOR CALLS
(LATOP%)

| | | |
|----------|--------|---|
| Function | Symbol | Meaning |
| 0 | .LASET | Set LAT parameters for local node. This function is used to set the dynamic parameters for the host in the local node. WHEEL or OPERATOR privileges are required. The argument block used to set the parameters is: |

| | | |
|------|--------|--|
| Word | Symbol | Contents |
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LASET |
| 2 | .LAPRM | Parameter number for parameter being set. The following parameters can be set: |

| | | |
|------|--------|---|
| Code | Symbol | Meaning |
| 1 | .LPMAC | Maximum number of active circuits |
| 2 | .LPMCO | Maximum number of simultaneous connects |
| 3 | .LPNUM | Host number |
| 4 | .LPLAS | LAT access state |
| 5 | .LPRLI | Circuit retransmit limit |
| 6 | .LPTIM | Circuit timer initial value |
| 7 | .LPMTI | Multicast timer initial value |
| 10 | .LPCOD | Group codes |
| 11 | .LPNNM | Host node name |
| 12 | .LPNID | Host node identification string |
| 13 | .LPSRV | Service rating and description |

TOPS-20 MONITOR CALLS
(LATOP%)

- 3 .LAVAL Contents depend on the parameter code:
- | Code | Contents |
|-------|--|
| 1-7 | New parameter value |
| 10 | Address of a bit mask representing codes to be set |
| 11-13 | ASCIZ string pointer to string representing parameters |
- 4 .LAQUA Required for parameter 13 only. Contains the following:
- | Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | LA%RAT | Set the rating as specified in the right half of this word. If all ones, the rating is set to DYNAMIC. |
| 1 | LA%DSC | Set the service description as specified in the next word. |
- If a particular bit is not set, the action taken depends on whether or not the service name previously existed: if previously existent, the parameter value is not changed. Otherwise the default for the parameter is set.
- 5 .LADSC An ASCIZ string pointer to the service description string to be set. If LA%DSC is set and this parameter is zero, the current service description is cleared.
- 1 .LACLRL Clear local node's LAT parameters. This function is used to clear the dynamic parameters for the host in the local node. WHEEL or OPERATOR privileges are required. The format of the argument block is:

TOPS-20 MONITOR CALLS
(LATOP%)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LACLR |
| 2 | .LAPRM | Parameter number for parameter to clear. Parameter numbers are the same as those defined for the .LASET function. Parameters 4 and 11 cannot be cleared. To change them, the .LASET function must be used. |
| 3 | .LAVAL | Depends on parameter code in .LAPRM. For parameter code 10, contains the address of the group code bit mask. For parameter 13, contains the ASCIZ pointer to service name to clear. This word is ignored for all other parameters. |

2 .LASCH Show the local node's LAT parameters. This function is used to show the dynamic, static, and permanent parameters for the host in the local node. The format of the argument block is:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LASCH |
| 2 | .LABCT | Number of words returned,,number of words reserved for returned information. |
| 3 | .LABFA | Address of location where information is stored upon return (show buffer). The format of the buffer returned to the user follows the function descriptions. |

3 .LASTC Show connects. This function is used to show all currently active LAT terminal connections at the local node. The format of the argument block is:

TOPS-20 MONITOR CALLS
(LATOP%)

| Word | Symbol | Contents | | | | | | |
|------|--------|--|-----|--------|---------|---|--------|---|
| 0 | .LAACT | Length of the argument block, including this word. | | | | | | |
| 1 | .LAFCN | .LASTC | | | | | | |
| 2 | .LABCT | Flags,,number of words reserved for returned information. On return, number of words returned,,number of words reserved. The following flag can be set: | | | | | | |
| | | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>LA%ECB</td> <td>Set to return information in Extended Connect Blocks. If not set, return standard Connect Blocks.</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | 0 | LA%ECB | Set to return information in Extended Connect Blocks. If not set, return standard Connect Blocks. |
| Bit | Symbol | Meaning | | | | | | |
| 0 | LA%ECB | Set to return information in Extended Connect Blocks. If not set, return standard Connect Blocks. | | | | | | |
| 3 | .LABFA | Address where information is returned (show buffer). The format of the buffer returned to the user follows the function descriptions. | | | | | | |

4 .LASAS Show Adjacent Servers. This function returns information about LAT servers that can access the local node. The format of the argument block is:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LASAS |
| 2 | .LABCT | Number of words returned,,number of words reserved for returned information. |
| 3 | .LABFA | Address where information is returned (show buffer). The format of the buffer returned to the user follows the function descriptions. |
| 4 | .LAQUA | ASCIZ string pointer to server name if information about a specific server is requested (returns full format server block). If this word is 0 (default), a summary of all |

TOPS-20 MONITOR CALLS
(LATOP%)

servers is returned (short form server block).

- 5 .LASCO Show Counters. Argument block format:
- | Word | Symbol | Contents |
|------|--------|--|
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LASCO |
| 2 | .LABCT | Number of words returned,,number of words reserved for returned information. |
| 3 | .LABFA | Address where information is returned (show buffer). The format of the buffer returned to the user follows the function descriptions. |
| 4 | .LAQUA | ASCIZ string pointer to server name if information about a specific server is requested (returns full format server block). If this word is 0 (default), a summary of all servers is returned (short form server block). |
- 6 .LAZCO Zero Counters. WHEEL or OPERATOR privileges are required. The format of the argument block is:
- | Word | Symbol | Contents |
|------|--------|--|
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LAZRO |
| 2 | .LABCT | unused |
| 3 | .LABFA | unused |
| 4 | .LAQUA | ASCIZ string pointer to server name if information about a specific server is requested (returns full format server block). If this word is 0 (default), a summary of all servers is returned (short form server block). |
- 7 .LARHC Request Host-Initiated Connect. This function requests a server to initiate a connection from an

TOPS-20 MONITOR CALLS
(LATOP%)

Application Terminal. If the connection completes successfully, the requesting process has an assigned TTY line to the Application Terminal. This function requires WHEEL or OPERATOR privileges. The format of the argument block is:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .LAACT | Length of the argument block, including this word. |
| 1 | .LAFCN | .LARHC |
| 2 | .LAPRM | Flags,,Connect-id. The following flags may be set: |

| Bit | Symbol | Meaning |
|-----|--------|---|
| 0 | LA%PSI | When set, the word .LAVAL should contain the PSI channel on which to interrupt the process when the connection is either made or rejected. If not set, the LATOP% JSYS block until either the connection is actually made, or the connection is rejected. |

If connection is made, the terminal designator can be obtained with the LATOP% function: .LASHC. A handle for use with the .LATHC and .LASHC functions is returned in LA%CID.

NOTE

When LA%PSI is set, you must have initialized the Software Interrupt

TOPS-20 MONITOR CALLS
(LATOP%)

| | | |
|-------|--------|--|
| | | System. (See Section 2.6 for more information on using Software Interrupts.) |
| 1 | LA%QUE | If set, request is queued for access to application terminal. If not set, request is immediately accessed to application terminal. |
| 3 | LA%JOB | Used by the .LASHC and .LATHC functions, and ignored by the .LARHC function. |
| 4-17 | | Unused - Reserved for DEC. |
| 18-35 | LA%CID | Connect-id returned for use with the .LATHC and .LASHC functions. |
| 3 | .LAVAL | If the LA%PSI flag is clear, this location returns the terminal designator if the connection has been made, or this location returns a reject code if the connection has been rejected. (For possible reject codes see below.) If the LA%PSI flag is set, this location should be set to the PSI channel number on which you wish to be interrupted. |
| 4 | .LASVR | Byte pointer to the Server Name (or zero). |
| 5 | .LASVC | Byte pointer to the Service Name (or zero). |
| 6 | .LAPRT | Byte pointer to the Port Name (or zero). |
| 8 | .LATHC | Terminate Host-Initiated Connect. This function |

TOPS-20 MONITOR CALLS
(LATOP%)

terminates connections from Application Terminals. The function requires WHEEL or OPERATOR privileges.

The argument block for the .LATHC function has the same format as the one used by the .LARHC function. To cancel a particular pending connect, you can use the same argument block by changing word .LAFCN from .LARHC to .LATHC. The format of the argument block is:

| Word | Symbol | Contents | | | | | | | | | | | | |
|-------|--------|---|-----|--------|---------|---|--------|--|------|--|----------------------------|-------|--------|--|
| 0 | .LAACT | Length of the argument block, including this word. | | | | | | | | | | | | |
| 1 | .LAFCN | .LATHC | | | | | | | | | | | | |
| 2 | .LAPRM | Flags,,Connect-id. The following flags may be set: | | | | | | | | | | | | |
| | | <table border="0"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>LA%JOB</td> <td>If set, terminate all pending requests for this job.</td> </tr> <tr> <td>4-17</td> <td></td> <td>Unused - reserved for DEC.</td> </tr> <tr> <td>18-35</td> <td>LA%CID</td> <td>If LA%JOB is not set, terminate the request associated with this Connect-id.</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | 3 | LA%JOB | If set, terminate all pending requests for this job. | 4-17 | | Unused - reserved for DEC. | 18-35 | LA%CID | If LA%JOB is not set, terminate the request associated with this Connect-id. |
| Bit | Symbol | Meaning | | | | | | | | | | | | |
| 3 | LA%JOB | If set, terminate all pending requests for this job. | | | | | | | | | | | | |
| 4-17 | | Unused - reserved for DEC. | | | | | | | | | | | | |
| 18-35 | LA%CID | If LA%JOB is not set, terminate the request associated with this Connect-id. | | | | | | | | | | | | |
| 3 | .LAVAL | Ignored | | | | | | | | | | | | |
| 4 | .LASVR | Ignored | | | | | | | | | | | | |
| 5 | .LASVC | Ignored | | | | | | | | | | | | |
| 6 | .LAPRT | Ignored | | | | | | | | | | | | |

9 .LASHC Show Host-Initiated Connects. This function returns information about connections from Application Terminals. The function information returned is in the form of a "Status Block" (see .LASHC Status Block format below). The format of the argument block is:

TOPS-20 MONITOR CALLS
(LATOP%)

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .LAACT | Length of the argument block, including this word. On return, the left half contains the number of words returned. |
| 1 | .LAFCN | .LASHC |
| 2 | .LABCT | The number of words reserved for returned information. |
| 3 | .LABFA | Address where information is returned (show buffer). |
| 4 | .LAQUA | Flags,,Connect-id. If LA%SYS is set in this word, return information about all Application Terminal connections on the system. If LA%JOB is set in this word, return information about all application terminal connections for this job. Otherwise, LA%CID contains the Connect-id of the request to return information. |

Within the .LARHC function, the possible .LAVAL reject codes are:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .LAUNK | Reason is unknown |
| 1 | .LAURD | User requested disconnect |
| 2 | .LASSP | System shutdown in progress |
| 3 | .LAISR | Invalid slot received |
| 4 | .LAISC | Invalid service class |
| 5 | .LAIRS | Insufficient resources to satisfy request |
| 6 | .LASIU | Service in use |
| 7 | .LANSS | No such service |
| 8 | .LASDI | Service is disabled |
| 9 | .LASNP | Service is not offered by requested port |
| 10 | .LANSP | No such port |
| 11 | .LAIPW | Invalid password |
| 12 | .LAENQ | Entry is not in the queue |
| 13 | .LAIAR | Immediate access rejected |
| 14 | .LAACD | Access denied |
| 15 | .LACSR | Corrupted solicit request |
| 16 | .LACTI | Command message type is illegal |
| 17 | .LASCS | Start slot can not be sent |
| 18 | .LAQED | Queue entry deleted by local node |
| 19 | .LAIRP | Inconsistent or illegal request parameters |

TOPS-20 MONITOR CALLS
(LATOP%)

With the .LARHC function, all combinations of Server Name, Service Name, and Port Name are defined as follows:

| Combination | Definition |
|---|--|
| Server Name only | Not Allowed |
| Service Name only | Not Allowed |
| Port Name only | Not Allowed |
| Service Name and Port Name | Not Allowed |
| Server Name and Port Name | Request a connection to a particular port on a particular server. |
| Server Name and Service Name | Request a connection to a particular service on a particular server. Note that a service can be offered on more than one port. |
| Server Name, Service Name, and Port Name | Request a connection to a particular port on a particular server if that port offers the requested service. |

SHOW BLOCK FORMATS

Several LATOP% functions return information in a buffer starting at the address stored in word .LABFA of the argument block. The functions and the format of the information returned are listed below.

.LASCH (Show characteristics)

Show buffer format is:

| 35 | 18 | 0 |
|---------------------|-------------------------|---|
| MAX_ALLOC_CIRCUITS | N_ALLOC_CIRCUITS | |
| MAX_ACTIVE_CIRCUITS | N_ACTIVE_CIRCUITS | |
| MAX_CONNECTS | N_CONNECTS | |
| HOST_NUMBER | LAT_TERMINAL_ACCESS_STA | |

TOPS-20 MONITOR CALLS
(LATOP%)

| | |
|--------------------------------------|----------------------|
| HOST_RETRANSMIT_LIMIT | HOST_CIRCUIT_TIMER |
| HOST_MULTICAST_TIMER | RESERVED |
| HI_PROTOCOL_VERSION | LO_PROTOCOL_VERSION |
| PROTOCOL_ECO | CUR_PROTOCOL_VERSION |
| MAX_SLOT_SIZE | MAX_SLOTS |
| FRAME_SIZE | MAX_SERVICES |
| HOST_GROUP_CODES (8 words) | |
| HOST_NAME count | HOST_IDENT count |
| HOST_NAME (2 words) | |
| HOST_IDENTIFICATION (13 words) | |
| Service Blocks (19 words/group name) | |

Service block format is:

| | |
|--------------------------------|-------------------------|
| HOST_SERVICE_NAME_RATING | |
| SERVICE_NAME count | SERVICE_DESCRIPTION cnt |
| SERVICE_NAME (4 words) | |
| SERVICE_DESCRIPTION (13 words) | |

.LASTC (Show connects)

There is one connect block returned for each LAT connection.

The connect block format is:

| |
|---------------------|
| Terminal Designator |
|---------------------|

TOPS-20 MONITOR CALLS
(LATOP%)

| | |
|-----------------------|---------------|
| Server Name Count | Indeterminate |
| Server Name (4 words) | |

The extended connect block format is: (LA%ECB is set)

| | |
|------------------------|--------------------|
| Terminal Designator | |
| Server Name Count | Port Type |
| Server Name (4 words) | |
| Port Name Count | Service Name Count |
| Port Name (4 words) | |
| Service Name (4 words) | |

The Server Name, Port Name, and Service Name are 7-bit ASCIZ strings. The Count fields do not include terminating nulls. The following values are defined for the Port Type:

| Value | Symbol | Meaning |
|-------|--------|---|
| 1 | .LATTY | This is a standard LAT terminal connection. |
| 2 | .LADLP | This is a dialup LAT terminal connection. |
| 3 | .LAAPP | This is a LAT application terminal. |

.LASAS (Show adjacent servers)

A full format block is returned when the .LASAS request specifies a server name in argument .LAQUA.

| | |
|-----------------------------------|------------------|
| Server Ethernet Address (2 words) | |
| FRAME_SIZE | SERVER_VERSION |
| MAX_SLOTS | indeterminate |
| CIRCUIT_TIMER | KEEP-ALIVE_TIMER |
| PRODUCT_TYPE | STATE |

TOPS-20 MONITOR CALLS
(LATOP%)

| | |
|---------------------------|-------------------|
| SERVER_NUMBER | SERVER_NAME count |
| SERVER_LOCATION count | unused |
| SERVER_NAME (4 words) | |
| SERVER LOCATION (4 words) | |

A short format block is returned when the .LASAS request specifies no server name.

| | |
|----------------------------|-------------------|
| SERVER_NUMBER | SERVER_NAME count |
| SERVER_NAME (4 words) | |
| ETHERNET_ADDRESS (2 words) | |

.LASCO (Show counters) and .LAZCO (Zero counters)

Counter Block Format:

| |
|---------------------------|
| Messages Received |
| Messages Sent |
| Messages Retransmitted |
| Receive Sequence Errors |
| Illegal Messages Received |
| Resource Failures |

.LASHC (Show Host-Initiated Connects) Status Block

Status block format is:

| | |
|------------|-------------|
| Job Number | Connect ID |
| Status | Queue Depth |

TOPS-20 MONITOR CALLS
(LATOP%)

| | |
|------------------------|-----------------|
| SERVER_NAME count | PORT_NAME count |
| SERVER_NAME (4 words) | |
| PORT_NAME (4 words) | |
| SERVICE_NAME count | Indeterminate |
| SERVICE_NAME (4 words) | |

Possible status values are:

| Value | Symbol | Meaning |
|---------------------|--------|-----------------------------|
| Terminal Designator | | Request was accepted. |
| Reject Code | | Request was rejected. |
| 377777 | .LASOL | Request is being solicited. |
| 377776 | .LAQUE | Request is being queued. |
| 377775 | .LACAN | Request has been canceled. |
| 377774 | .LATMO | Request has timed out. |

Generates an illegal instruction trap on failure.

LATOP% ERROR MNEMONICS:

```

ARGX02: Invalid function
ARGX04: Argument block too small
ARGX05: Argument block too long
CAPX1: WHEEL or OPERATOR capability required
LATX01: Buffer size too small for available data
LATX02: LAT parameter value out of range
LATX03: LAT is not operational
LATX04: Invalid or unknown LAT server name
LATX05: Invalid LAT parameter
LATX06: Invalid LAT parameter value
LATX07: Invalid or unknown LAT service name
LATX08: Insufficient LAT Resources
LATX09: LAT Host name already set
LATX10: Invalid or unknown LAT port name
LATX11: Invalid or unknown connect id

```

TOPS-20 MONITOR CALLS
(LGOUT)

Kills the specified job and appends an accounting entry to the accounting data file. However, no entry is appended if the job was never logged in (that is, a CTRL/C was typed, but no login occurred).

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Number of the job to be logged out, or -1 for the current job

RETURNS +1: Failure, error code in AC1

+2: Success

When a specific job number is given in AC1, it must refer to either a PTY job controlled by the current job or a job logged in under the same user name as the current job. Otherwise, to give a specific job number, the process must have WHEEL or OPERATOR capability enabled. An argument of -1 must be given if the current job wishes to kill itself (that is, the job number given cannot be the same as the current job). Note that this monitor call does not return if the argument in AC1 is -1.

The LGOUT monitor call outputs the time used (both CPU and console), the job number, the current date and time, and the name of the user who logged out the job if it is not the calling job. This information is output on the terminal to which the job being logged out is attached.

LGOUT ERROR MNEMONICS:

LOUTX1: Illegal to specify job number when logging out own job

LOUTX2: Invalid job number

LOUTX3: WHEEL or OPERATOR capability required

LOUTX4: LOG capability required

LOUTX5: Illegal to log out job 0

TOPS-20 MONITOR CALLS
(LLMOP%)

NOTE

This JSYS is primarily intended for system use. The information returned may change in a future release.

Provides access to Network Interconnect (NI) Remote Console Service and performs Ethernet loopback operations.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code

AC2: Argument block

RETURNS +1: Always

Interface to NI Loopback Requestor/Server

This interface provides three basic functions: checking the status of pending requests, initiating requests, and enabling to read unsolicited datagrams. The functions listed below perform the actual Ethernet loopback operations.

All loopback operations are performed with padding enabled for the loopback protocol portal.

| Function | Symbol | Meaning |
|----------|--------|--|
| 0 | .ELDIR | Builds an Ethernet loopback message from data supplied in the argument block, and transmits to the destination address. The argument block is: |

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .LMCID | Channel ID. B34-35 (LM%CID) contain the value (from 0-3) of the Ethernet port to use. |
| 1-2 | .LMDST | Destination address. |
| 3 | .LMREQ | Request number, containing: |

| Bit | Symbol | Meaning |
|-----|--------|--|
| 0 | LM%AIC | Assigns interrupt channel specified in LM%ICH if this flag is set; if off, the LM%ICH field is |

TOPS-20 MONITOR CALLS
(LLMOP%)

ignored and no interrupts are given.

12-17 LM%ICH Interrupt channel number. Contains number of PSI channel to interrupt when loopback reply message arrives from remote system.

18-35 LM%REQ Contains request number returned by LLMOP%. This value is used in function .ELRPY, .ELABT, .ELSTS.

4 .LMRBL Loopback request data buffer length. Bits 18-35 (LM%MBL) contain the length of the data protion of the loopback message.

5 .LMRBP Pointer to loopback request data buffer.

1 .ELAST Builds an Ethernet loopback message from data supplied in the argument block, and transmits it according to the type of assistance requested. Argument block words 0-5, .LMCID, .LMDST, .LMREQ, .LMRBL, and .LMRBP, are described in function .ELDIR. The remainder of the argument block is:

Word Symbol Contents

6-7 .LMAST Address of the node used as the assistant in the loopback request. This cannot be a multicast address.

10 .LMHLP Assistance level

Level Symbol Meaning

1 .LMXMT Transmit. Forwards the loopback message to destination and local nodes.

2 .LMRCV Receive. Forwards the loopback message to assistant and local nodes.

TOPS-20 MONITOR CALLS
(LLMOP%)

| | | | | | |
|---|--------|--|--------|--|--|
| | | | 3 | .LMFUL | Full. Forwards the loopback message to destination, assistant and local nodes. |
| 2 | .ELRPY | Reads loopback reply. The format of the argument block is: | | | |
| | | Word | Symbol | Contents | |
| | | 0 | .LMCID | Channel ID. Bits 34 and 35 (LM%CID) contain the value of the Ethernet port to use. | |
| | | 1-2 | .LMSRC | Upon return, contains address of the remote system that satisfied the loop assisted operation. | |
| | | 3 | .LMREQ | Request number. Bits 18-35 (LM%REQ) contain the request number of the reply to be read. The caller is blocked until the reply arrives. | |
| | | 4 | .LMRBL | Loop response buffer length. Upon return, bits 0-17 (LM%RML) contain the length of the received loop reply message data. Bits 18-35 hold the maximum length of the loop response data buffer (supplied by user). | |
| | | 5 | .LMRBP | Pointer to loop reply buffer. | |
| 4 | .ELABT | Aborts Ethernet loop request. The format of the argument block is: | | | |
| | | Word | Symbol | Contents | |
| | | 0 | .LMCID | Channel ID. Bits 34-35 (LM%CID) contain the value of the Ethernet port to use. | |
| | | 3 | .LMREQ | Request number. Bits 18-35 (LM%REQ) contain the number of the request to be aborted. | |
| 5 | .ELSTS | Obtains the status of Ethernet loopback requests. The format of the argument block is: | | | |

TOPS-20 MONITOR CALLS
(LLMOP%)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .LMCID | Channel ID. Bits 34-35 contain the value of the Ethernet port to use. |
| 1 | .LMSTF | Upon return, contains status code for the request. Bits 18-35 (LM%RTC) contain one of the following status return codes: |
| | | Code Symbol Meaning |
| | 0 | .LMPND Request pending, not complete. |
| | 1 | .LMSUC Request completed successfully. |
| 3 | .LMREQ | Request number. Bits 18-35 (LM%REQ) contain the number of the request assigned by function .ELDIR or function .ELAST. |

Interface to NI Remote Console

This interface provides four basic functions; gaining access to the NI Remote Console Service, initiating a request, checking the status of a pending request, and enabling to read unsolicited datagrams.

LLMOP% provides the following remote console functions:

| Function | Symbol | Meaning |
|----------|--------|--|
| 6 | .RCRID | Transmits a Read Identity protocol message to the destination address node on the Ethernet. Function .RCRPY must be used to read the system ID reply message. This function does not block the issuing process. The format of the argument block is: |

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .LMCID | Channel ID. Bits 34-35 contain the value of the Ethernet port to use. |
| 1-2 | .LMDST | Destination address. |
| 3 | .LMREQ | Request number, containing: |

TOPS-20 MONITOR CALLS
(LLMOP%)

| | Bit | Symbol | Meaning | |
|----|-------|--------|--|---|
| | 0 | LM%AIC | Assigns interrupt channel specified in LM%ICH if this flag is set; if off, the LM%ICH field is ignored and no interrupts are given. | |
| | 12-17 | LM%ICH | Interrupt channel number. Contains number of PSI channel to interrupt when loopback reply message arrives from remote system. | |
| | 18-35 | LM%REQ | Contains request number returned by LLMOP%. This value must be used in functions .RCRPY, .RCABT, and .RCSTS. | |
| 7 | | .RCRCT | Transmits a Read Counters protocol message to the destination address node on the Ethernet. Use function .RCRPY to read the System ID reply message. The argument block is identical to that of function .RCRID. | |
| 11 | | .RCRBT | Transmits a Boot protocol message to the destination address node on the Ethernet. This function blocks the issuing process until the transmit completes. The format of the argument block is: | |
| | | Word | Symbol | Contents |
| | | 0 | .LMCID | Channel ID. Bits 34-35 (LM%CID) contain the value of the Ethernet port to use. |
| | | 1-2 | .LMDST | Destination address. |
| | | 3-4 | .LMPWD | 8-byte password verification code transmitted to the remote system for its use in deciding whether to allow the boot request. |
| | | 5 | .LMCIF | Control information, in the form: |

TOPS-20 MONITOR CALLS
(LLMOP%)

| | | Bit | Symbol | Meaning |
|----|--------|---|--------|--|
| | | 26 | LM%BDV | Boot device. 0 = system default; 1 = specified device. |
| | | 27 | LM%BSV | Boot server. 0 = system default; 1 = requesting system. |
| | | 28-35 | LM%PRO | Processor to boot. 0 = system processor; 1 = communication processor. |
| | | 6 | .LMDID | Device ID in an 8-bit byte string. |
| | | 7 | .LMSID | Software ID in an 8-bit byte string. |
| 12 | .RCRPY | Reads the response to a .RCRID (request ID) or .RCRCT (request counters) function. The format of the argument block is: | | |
| | | Word | Symbol | Contents |
| | | 0 | .LMCID | Channel ID. If B0(LM%MRF) is set, there are more replies available for this request. Bits 34-35 contain the value of the Ethernet port to use. |
| | | 1-2 | .LMSRC | Address of responding node. |
| | | 3 | .LMREQ | Request number. Bits 18-35 (LM%REQ) contain the request number of the reply to be read. The caller is blocked until the reply arrives. |
| | | 4 | .LMRBL | Console response buffer length. Upon return, bits 0-17 (LM%RML) contain the length of the received console reply message data. Bits 18-35 hold the maximum length of the remote console response data buffer (supplied by user). |
| | | 5 | .LMRBP | Pointer to console reply buffer. |

TOPS-20 MONITOR CALLS
(LLMOP%)

- 13 .RCRSV Transmits a reserve remote console MOP message. The argument block contains words .lmCID, .lmDST, and .lmPWD, as described for function .RCRBT.
- 14 .RCREL Transmits a release remote console MOP message. The argument block contains words .lmCID and .lmDST, as described for function .RCRBT.
- 15 .RCSND Sends ASCII console command data to remote console and polls for response data. If no command data is included, this function only polls for response data. The format of the argument block is:

| Word | Symbol | Contents | | | | | | |
|-------|--------|--|-----|--------|---------|-------|--------|--|
| 0 | .LMCID | Channel ID | | | | | | |
| | | <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>34-35</td> <td>LM%CID</td> <td>Channel ID. Value specifying Ethernet port to use.</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | 34-35 | LM%CID | Channel ID. Value specifying Ethernet port to use. |
| Bit | Symbol | Meaning | | | | | | |
| 34-35 | LM%CID | Channel ID. Value specifying Ethernet port to use. | | | | | | |
| 1-2 | .LMDST | Destination address. | | | | | | |
| 3 | .LMREQ | Request number, as described for function .RCRID. | | | | | | |
| 4 | .LMRBL | Length of console request buffer. Bits 18-35 (LM%MBL) contain the maximum buffer length. | | | | | | |
| 5 | .LMRBP | Pointer to remote console data buffer. | | | | | | |

- 16 .RCPOL Polls for completion of function .RCSND (send console command). The format of the argument block is:

| Word | Symbol | Meaning | | | | | | |
|-------|--------|---|-----|--------|---------|-------|--------|-------------|
| 0 | .LMCID | Channel ID | | | | | | |
| | | <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>34-35</td> <td>LM%CID</td> <td>Channel ID.</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | 34-35 | LM%CID | Channel ID. |
| Bit | Symbol | Meaning | | | | | | |
| 34-35 | LM%CID | Channel ID. | | | | | | |
| 1-2 | .LMSRC | Address of node that sent this reply. | | | | | | |
| 3 | .LMREQ | Request number. Bits 18-35 | | | | | | |

TOPS-20 MONITOR CALLS
(LLMOP%)

(LM%REQ) contain the request ID assigned by function .RCSND.

| | | |
|----|--------|---|
| 4 | .LMRBL | Length of console response buffer. Same as described for function .RCRPY. |
| 5 | .LMRBP | Pointer to remote console data buffer. |
| 17 | .RCAIC | Assigns software interrupt channel for Ethernet remote console message. The format of the argument block is: |
| | Word | Symbol Contents |
| | 0 | .LMCID Channel ID. Bits 34-35 (LM%CID) contain the value of the Ethernet channel to use. |
| | 1 | .LMICF Interrupt channel flags. |
| | | Bit Symbol Meaning |
| | | 0 LM%AIC Assigns interrupt channel specified in LM%ICH if set; if off, the channel is deassigned. |
| | | 12-17 LM%ICH Contains PSI channel to interrupt when remote console reply message arrives. This function returns an error for all but the first process to request it. |
| 20 | .RCABT | Aborts an outstanding remote console request. The format of the argument block is the same as described for function .ELABT. |
| 21 | .RCSTS | Obtains status of a remote console request. The format of the argument block is the same as described for function .ELSTS. |
| 22 | .RCADR | Obtains a channel address. The format of the argument block is: |

TOPS-20 MONITOR CALLS
(LLMOP%)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .LMCID | Channel ID. Bits 34-35 (LM%CID) contain the value of the Ethernet port to use. |
| 1-2 | .LMHWA | Hardware address. |
| 3-4 | .LMPYA | Physical address. |

LLMOP% ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required
 ARGX02: Invalid function
 LLMX01: Transmit Datagram Failed
 LLMX02: LLMOP State is OFF
 LLMX03: Invalid byte pointer
 LLMX04: Nonexistent Request Number
 LLMX05: Invalid KLNI channel specified
 LLMX06: Configurator interrupts assigned to another process
 LLMX99: LLMOP Internal Error
 ARGX13: Invalid software interrupt channel number

Translates a logical name to its original definition string. (See Section 2.2.2 and the CRLNM and INLNM monitor calls descriptions.)

ACCEPTS IN AC1: Function code

AC2: Pointer to the logical name. The logical name must not contain a terminating colon.

AC3: Pointer to the string where the original logical name definition is to be written. The name returned includes a terminating colon.

RETURNS +1: Failure, error code in AC1

+2: Success, updated string pointer in AC3

The codes for the functions are as follows:

0 .LNSJB Obtain the job-wide definition of the logical name.
 1 .LNSSY Obtain the system definition of the logical name.

TOPS-20 MONITOR CALLS
(LNMST)

LNMST ERROR MNEMONICS:

GJFX22: Insufficient system resources (Job Storage Block full)
LNSTX1: No such logical name
LNSTX2: Invalid function

Logs a job into the system. Useful for logging in from an idle terminal on which a CTRL/C has been typed.

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: 36-bit user number under which user will log in

AC2: Pointer to beginning of password string

AC3: Account number in bits 3-35 if bits 0-2 are 5. Otherwise contains a pointer to an account string. If a null byte is not seen, the string is terminated after 39 characters are

RETURNS: +1: Failure, error code in AC1

+2: Success with:

AC1: Date and time of last interactive login

AC2: Date and time of last non-interactive login

AC3: Password expiration date (0 if none, -1 if this is the last time a user can login - that is, if the password has expired)

AC4: Number of interactive login failures, ,number of non-interactive login failures

The LOGIN% monitor call will allow 1 login after the user's password has expired. It is the user's responsibility to then change the password.

The LOGIN monitor call does not require a password if the controlling terminal is a pseudo-terminal and the controlling job either has the WHEEL or OPERATOR capability enabled or is logged in as the same user being logged in for this job.

TOPS-20 MONITOR CALLS
(LOGIN)

If the call is successful, an accounting entry is appended to the accounting data file. If the account validation facility is enabled, the LOGIN call verifies either the account given or the default account of the user being logged in.

LOGIN ERROR MNEMONICS:

LGINX1: Invalid account identifier
LGINX2: Directory is "files-only" and cannot be logged in to
LGINX3: Internal format of directory is incorrect
LGINX4: Invalid password
LGINX5: Job is already logged in
LGINX6: No more job slots available for logging in

Loads the direct access Vertical Formatting Unit (VFU) or translation Random Access Memory (RAM) for the line printer. This call is executed at system startup by the program that configures the system.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: JFN of file containing VFU or RAM

AC2: Status bits in the left half, and function code in the right half

AC3: Unit number of line printer

RETURNS +1: Always

The following status bit is currently defined.

B0(MO%LCP) Line printer is a lowercase printer.

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 32 | .MOLVF | Load the VFU from the file indicated by the given JFN. |
| 34 | .MOLTR | Load the translation RAM from the file indicated by the given JFN. |

TOPS-20 MONITOR CALLS
(LPINI)

The line printer must not be opened by any process when this call is executed. If a condition occurs that prevents the VFU or RAM from being loaded (for example, the line printer is off line), the name of the file will be stored. The VFU or RAM will then be loaded automatically the next time a process performs output to the line printer.

Generates an illegal instruction interrupt on error conditions below.

LPINI ERROR MNEMONICS:

LPINX1: Invalid unit number
LPINX2: WHEEL or OPERATOR capability required
LPINX3: Illegal to load RAM or VFU while device is OPEN

Transfers control to the MDDT program while preserving the context of the process that issued the MDDT% JSYS. The terminal keyboard is activated and the user may enter commands to the MDDT program, or may return to TOPS-20 command level by typing CTRL/C, or may return to the issuing process by typing CTRL/Z.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

The MDDT% JSYS accepts no arguments.

MDDT% ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

Returns the value of the execution accounting meter or the memory reference accounting meter. These values do not represent time as in "clock time"; rather, they represent the amount of time that the EBOX was busy and how many times the MBOX was referenced by the EBOX.

TOPS-20 MONITOR CALLS
(METER%)

ACCEPTS IN AC1: Function code

RETURNS +1: Always, with 59-bit value in AC2 and AC3

Function Codes:

| Code | Symbol | Meaning |
|------|--------|---|
| 1 | .MERE | Read process execution accounting meter doubleword. Value returned is EBOX busy time (number of EBOX ticks). |
| 2 | .MERMA | Read process memory-reference accounting meter doubleword. Value returned is count of MBOX references (number of MBOX ticks). |

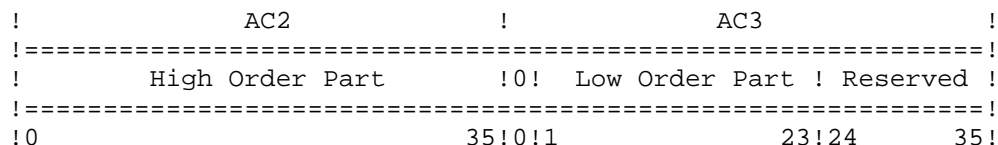
The accounting meters have bits that allow executive PI overhead and executive non-PI overhead to be included in the doubleword count. These are turned off by default (the monitor must be rebuilt to set them), so (by default) the EBOX count does not include the monitor overhead of paging, scheduling, or swapping. The EBOX count primarily includes only the EBOX time spent executing the instructions and JSYSs in the user's program.

Interrupts caused by IO, paging, swapping, and so on, can cause instruction restarts or require pager refills, and these are included in the count. Because these interrupts depend on a variety of system variables, such as load average, subsequent timings of the same event will return varying count values. These fluctuations can be "smoothed" by timing the event repeatedly and taking the average of the values returned.

The MBOX reference count has the same specifications as the EBOX count, and is subject to the same kind of fluctuations. Cache hit/no hit introduces an additional source of fluctuations. Again, timing the event repeatedly and taking the average of the values returned will "smooth" the counts.

An event can be timed by an initial execution of METER%, a DMOVEM instruction to save the start value, and (after the event) a second execution of METER% followed by a DSUB instruction to find the elapsed number of ticks. For added accuracy, the average overhead for the timing sequence can be determined and subtracted from the average count value for the timed interval.

The following diagram illustrates the format of the value returned:



TOPS-20 MONITOR CALLS
(METER%)

Note that the following instruction changes the format of the values returned by the METER% call to form a right-justified doubleword value in AC2 and AC3.

ASHC AC2,-^D12

METER% ERROR MNEMONICS:

ARGX02: Invalid function code
METRX1: METER% not implemented for this processor

Retrieves an IPCF (Inter-Process Communication Facility) message from the process's input queue. See the Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

RESTRICTIONS: Some functions require WHEEL, OPERATOR or IPCF capability enabled.

ACCEPTS IN AC1: Length of packet descriptor block

AC2: Address of packet descriptor block

RETURNS +1: Failure, error code in AC1
+2: Success. The packet is retrieved and placed into the block indicated by word .IPCFP of the packet descriptor block. AC1 contains the length of the next entry in the queue in the left half and the flags from the next packet in the right half. This returned word is called the associated variable of the next entry in the queue. If the queue is empty, AC1 contains 0.

The format of the packet descriptor block is as follows:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .IPCFL | Flags. (See the MSEND call description.) If bit IP%CFB is set in this word, MRECV does not block until a packet is read. |
| 1 | .IPCFS | PID of sender. The caller does not supply this PID; the system fills it in when the packet is retrieved. |

TOPS-20 MONITOR CALLS
(MRECV)

| | | |
|----|---------|--|
| 2 | .IPCFR | PID of receiver. This PID can be one of three values: a specific PID, -1 to retrieve messages for any PID belonging to this process, or -2 to retrieve messages for any PID belonging to this job. When -1 or -2 is supplied, messages are not retrieved in any particular order except that messages from a specific PID are returned in the order in which they were received. |
| 3 | .IPCFFP | Pointer to block where message is to be placed (length of message in the left half and starting address of message in the right half). |
| 4 | .IPCFFD | User number of sender. Supplied by the monitor. |
| 5 | .IPCFFC | Enabled capabilities of sender. Supplied by the monitor. |
| 6 | .IPCFFS | Directory number of sender's connected directory. Supplied by the monitor. |
| 7 | .IPCFFA | Account string of sender. The caller supplies a pointer to the block where the account is to be placed. |
| 10 | .IPCFFL | Byte pointer to area to store logical location (node name) of sender. |

The caller (receiver) does not supply the information in words 4 through 7; the system fills in the words when the packet is retrieved. These words describe the sender at the time the message was sent and permit the receiver to validate messages. If a byte pointer is supplied in word .IPCFFL, the monitor will use it to return the ASCII string for the logical location of the sender.

See the MSEND call description for the flags that can be set in word .IPCFFL of the packet descriptor block.

MRECV ERROR MNEMONICS:

| | |
|----------|---|
| IPCFX1: | Length of packet descriptor block cannot be less than 4 |
| IPCFX2: | No message for this PID |
| IPCFX3: | Data too long for user's buffer |
| IPCFX4: | Receiver's PID invalid |
| IPCFX5: | Receiver's PID disabled |
| IPCFF11: | WHEEL or IPCF capability required |
| IPCFF14: | No PID's available to this job |
| IPCFF15: | No PID's available to this process |
| IPCFF16: | Receive and message data modes do not match |
| IPCFF24: | Invalid message size |

TOPS-20 MONITOR CALLS
(MRECV)

IPCF25: PID does not belong to this job
IPCF26: PID does not belong to this process
IPCF27: PID is not defined
IPCF28: PID not accessible by this process
IPCF29: PID already being used by another process
IPCF31: Invalid page number
IPCF32: Page is not private
IPCF34: Cannot receive into an existing page
IPCF36: PID not assigned on this LCS processor

Sends an IPCF (Inter-Process Communication Facility) message. The message is in the form of a packet and can be sent to either the specified PID or the system process <SYSTEM>INFO. See the TOPS-20 Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

RESTRICTIONS: Some functions require WHEEL, OPERATOR, or IPCF capability enabled.

ACCEPTS IN AC1: Length of packet descriptor block

AC2: Address of packet descriptor block

RETURNS +1: Failure, error code in AC1

+2: Success. The packet is sent to the receiver's input queue. Word .IPCFS of the packet descriptor block is updated with the sender's PID. This updating is done in case the PID was being defaulted or created by this call.

The format of the packet descriptor block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .IPCFL | Flags. (See below.) |
| 1 | .IPCFS | PID of sender; or address of PID if IP%CFR is set in WORD .IPCFL; or 0 if no PID exists for sender. This word will be filled in by the monitor if the caller is creating a PID (flag bit IP%CPD is on). |

TOPS-20 MONITOR CALLS
(MSEND)

- 2 .IPCFR PID of receiver, or 0 if receiver is <SYSTEM>INFO.
- 3 .IPCFL Pointer to message block (length of message in the left half and starting address of message in the right half). When a packet is sent to <SYSTEM>INFO, the message block contains the request being made. (See below.)

The following flags are defined in word .IPCFL of the packet descriptor block. These flags can be set on both the MSEND and MRECV calls.

Flags Set By Caller

- B0(IP%CFB) Do not block process if there are no messages in the queue. If this bit is set, an error is given if there are no messages.
- B1(IP%CFS) Use, as the sender's PID, the PID obtained from the address specified in word .IPCFS. Setting bit IP%CFS notifies the monitor that word .IPCFS contains an address, and the sender's PID is located at that address.
- B2(IP%CFR) Use, as the receiver's PID, the PID obtained from the address specified in word .IPCFR. Setting bit IP%CFR notifies the monitor that word .IPCFR contains an address, and the receiver's PID is located at that address.
- B3(IP%CFQ) Allow one send request above the quota. (The default send quota is 2.)
- B4(IP%TTL) Truncate the message, if it is larger than the space reserved. If this bit is not set, an error is given if the message is too large.
- B5(IP%CPD) Create a PID to use as the sender's PID and return it in word .IPCFS of the packet descriptor block. If flag IP%CFS is set, this function returns the created PID in the word to which the contents of .IPCFS points.
- B6(IP%JWP) Make the created PID be job wide (permanent until the job logs out). If this bit is not set, the PID is temporary until the process executes the RESET monitor call. If B5(IP%CPD) is not set, B6 is ignored.
- B7(IP%NOA) Do not allow other processes to use the created PID. If B5(IP%CPD) is not set, B7 is ignored.
- B8(IP%MON) Reserved for DIGITAL.

TOPS-20 MONITOR CALLS
(MSEND)

B18(IP%CFP) The packet is privileged. (This bit can be set only by a process with IPCF capability enabled.) When a privileged sender sets this bit, the MRECV and MUTIL calls return it set for any reply. An error is given if this bit is set by the sender and the receiver is not privileged.

B19(IP%CFV) The packet is a page of data. Word .IPCFP of the packet descriptor block contains 1000 in the left half and the page number in the right half. The page the packet is being sent to must be private.

B21(IP%INT) Reserved for DIGITAL.

B22(IP%EPN) Page number in word .IPCFP of the packet descriptor block is 18 bits long.

NOTE

When a process sends a page of data with MSEND, that page is removed from the process's map.

Flags Returned After Call

B20(IP%CFZ) A zero-length message was sent, and the packet consists of only the packet descriptor block.

B24-29(IP%CFE) Error code field for errors encountered by <SYSTEM>INFO during a send or receive request.

| Code | Symbol | Meaning |
|------|--------|-----------------------------------|
| 15 | .IPCPI | insufficient privileges |
| 16 | .IPCUF | invalid function |
| 67 | .IPCSN | <SYSTEM>INFO needs name |
| 72 | .IPCFF | <SYSTEM>INFO free space exhausted |
| 74 | .IPCBP | PID has no name or is invalid |
| 75 | .IPCDN | duplicate name has been specified |
| 76 | .IPCNN | unknown name has been specified |
| 77 | .IPCEN | invalid name has been specified |

B30-32(IP%FCF) System and sender code. This code can be set only by a process with IPCF capability enabled. The system returns the code so that a nonprivileged user can examine it.

| Code | Symbol | Meaning |
|------|--------|----------------------------------|
| 1 | .IPCCC | sent by <SYSTEM>IPCF |
| 2 | .IPCCF | sent by system-wide <SYSTEM>INFO |

TOPS-20 MONITOR CALLS
(MSEND)

3 .IPCCP sent by receiver's <SYSTEM>INFO
4 .IPCCG sent by system for QUEUE% JSYS

B33-35(IP%CFM) Field for return of special messages. This field can be set only by a process with WHEEL capability enabled. The system returns the information so that a nonprivileged user can examine it.

| Code | Symbol | Meaning |
|------|--------|--|
| 1 | .IPCFN | Process's input queue contains a packet that could not be delivered to intended PID. |

When the MSEND call is used to send a packet to <SYSTEM>INFO, the message portion of the packet (the first three words) contains the request. This request has the following format:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .IPCI0 | User-defined code in the left half and the function (see below) <SYSTEM>INFO is to perform in the right half. The user-defined code is used to associate the response from <SYSTEM>INFO with the appropriate request. |
| 1 | .IPCI1 | PID that is to receive a duplicate of the response from <SYSTEM>INFO. If this word is 0, the response is sent only to the originator of the request. |
| 2 | .IPCI2 | Argument for the requested function. (See below.) |

The functions that can be requested of <SYSTEM>INFO, along with their arguments, are as follows:

| Function | Argument | Meaning |
|----------|------------------|---|
| .IPCIW | name | Return the PID associated with the specified name. The PID is returned in word .IPCI1. |
| .IPCIG | PID | Return the name associated with the specified PID. The name is returned in word .IPCI1. |
| .IPCII | name in ASCIZ | Assign the specified name to the PID belonging to the process making the request. The temporary or permanent status of the PID is specified by flag bit IP%JWP(B6) when the PID was originally created. |
| .IPCIJ | name in ASCIZ | Identical to the .IPCII function. |

TOPS-20 MONITOR CALLS
(MSEND)

| | | |
|--------|-----|--|
| .IPCIK | PID | Inform a PID when certain other PID's are deleted. The PID to be "watched" for deletion is placed in word .IPCI2. When that PID is deleted, SYSTEM INFO sends a message to the requesting PID with .IPCKM in the IP%CFE field, and the deleted PID in word .IPCI0 of the message. This function requires WHEEL or OPERATOR privileges. |
| .IPCIS | PID | Disassociates all PIDs with names. However, the PID remains. To delete PID, use the .MUCHO and .MUDES functions of the MUTIL monitor call. This function (.IPCIS) requires WHEEL or OPERATOR capability enabled. |

MSEND ERROR MNEMONICS:

| | |
|---------|---|
| IPCFX1: | Length of packet descriptor block cannot be less than 4 |
| IPCFX4: | Receiver's PID invalid |
| IPCFX5: | Receiver's PID disabled |
| IPCFX6: | Send quota exceeded |
| IPCFX7: | Receiver quota exceeded |
| IPCFX8: | IPCF free space exhausted |
| IPCFX9: | Sender's PID invalid |
| IPCF11: | WHEEL or IPCF capability required |
| IPCF12: | No free PID's available |
| IPCF13: | PID quota exceeded |
| IPCF14: | No PID's available to this job |
| IPCF15: | No PID's available to this process |
| IPCF19: | No PID for [SYSTEM]INFO |
| IPCF24: | Invalid message size |
| IPCF25: | PID does not belong to this job |
| IPCF26: | PID does not belong to this process |
| IPCF27: | PID is not defined |
| IPCF28: | PID not accessible by this process |
| IPCF29: | PID already being used by another process |
| IPCF31: | Invalid page number |
| IPCF32: | Page is not private |
| IPCF36: | PID not assigned on this LCS processor |

Starts a process in monitor mode. This call allows job 0 to create multiple processes for handling various asynchronous monitor tasks.

TOPS-20 MONITOR CALLS
(MSFRK)

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled, or execution from monitor mode.

ACCEPTS IN AC1: Process handle

AC2: 36-bit PC word, with user mode and other flags in the left half and the virtual address in the right half

RETURNS +1: Always

Because the starting context of the process is undefined, the process being started should execute the following sequence of instructions at its starting address:

```
FBGN:  MOVSI 1,UMODF    ;fake user PC
        MOVEM 1,FPC     ;simulate the JSYS call
        MCENTR         ;establish usual top-level JSYS context
```

Generates an illegal instruction interrupt on error conditions below.

MSFRK ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
CAPX1: WHEEL or OPERATOR capability required

Performs various structure-dependent functions. These functions include mounting and dismounting structures, incrementing and decrementing mount counts for structures, and setting and obtaining the status of structures.

For regulated structures, the mount count must be incremented before access rights or JFNs can be given. All structures are regulated by default except the public structure or any structure declared non-regulated with the .MSSSS function of MSTR.

Some functions require a structure device designator as an argument. Use the STDEV JSYS to obtain a device designator for a structure.

RESTRICTIONS: Some functions require WHEEL, OPERATOR, or MAINTENANCE capability enabled.

TOPS-20 MONITOR CALLS
(MSTR)

ACCEPTS IN AC1: Length of the argument block in the left half and function code in the right half

AC2: Address of the argument block

RETURNS +1: Always, with some functions returning data in the argument block. (See individual function descriptions below.)

The available functions are summarized below.

| Function | Symbol | Privileged | Meaning |
|----------|--------|------------|---|
| 0 | .MSRNU | Yes | Return the status of the next disk unit. |
| 1 | .MSRUS | Yes | Return the status of the given disk unit. |
| 2 | .MSMNT | Yes | Mount the given structure. |
| 3 | .MSDIS | Yes | Dismount the given structure. |
| 4 | .MSGSS | No | Return the status of the given structure. |
| 5 | .MSSSS | Yes | Change the status of the given structure. |
| 6 | .MSINI | Yes | Initialize the given structure. |
| 7 | .MSIMC | No | Increment the mount count for the given structure for the job. |
| 10 | .MSDMC | No | Decrement the mount count for the given structure for the job. |
| 11 | .MSGSU | No | Return the job numbers of the users of the given structure. |
| 12 | .MSHOM | Yes | Modify the home block of the given structure. |
| 13 | .MSICF | No | Increment the mount count for the given structure for the given fork. |

TOPS-20 MONITOR CALLS
(MSTR)

| | | | |
|----|--------|-----|---|
| 14 | .MSDCF | No | Decrement the mount count for the given structure for the given fork. |
| 15 | .MSOFL | Yes | Receive interrupt when disk comes on-line. |
| 16 | .MSIIC | Yes | Ignore increment check for structure use |
| 17 | .MSCSM | Yes | Change structure mount attribute (CFS-20) |

Obtaining the Status of the Next Disk Unit - .MSRNU

This function returns the status of the next disk unit on the system. The next disk unit is determined by searching the current channel and looking for the next physical unit on that channel.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

The .MSRNU function accepts the channel, controller, and unit numbers in the first three words of the argument block. The time this function is executed, the value for each of these numbers is -1. After successful completion of this function, the channel, controller, and unit numbers are updated, and the software information about the disk drive is returned in the argument block. To locate all drives available for mounting structures, the channel, controller, and unit numbers returned from one .MSRNU function call are supplied on the next one until all units on all channels have been searched. When all units have been searched, the MSTR monitor call returns error MSTX18.

The format of the argument block, whose length is .MSRLN, is as follows:

| Word | Symbol | Meaning |
|------|------------|--|
| 0 | .MSRCH | Channel number (0-7) |
| 1 | .MSRCT | Controller number |
| 2 | .MSRUN | Unit number (0-7) |
| 3 | .MSRST | Returned software status of unit. The following status bits are defined: |
| | B0(MS%MNT) | Unit is part of a mounted structure |
| | B2(MS%DIA) | Unit is being used by an on-line diagnostic program |
| | B3(MS%OFL) | Unit is off line |

TOPS-20 MONITOR CALLS
(MSTR)

B4(MS%ERR) Unit has an error that was detected during reading

B5(MS%BBB) Unit has a bad BAT block. If this bit is on, the data returned word .MSRSN (word 4) and in words .MSRNS through .MSRFI (words 6 through 20) is indeterminate.

B6(MS%HBB) Unit has a bad HOME block

B7(MS%WLK) Unit is write locked

B8(MS%2PT) Unit is potentially dual-ported between systems

B9-17
(MS%TYP) Type of disk unit

| | | |
|----|--------|------|
| 1 | .MSRP4 | RP04 |
| 5 | .MSRP5 | RP05 |
| 6 | .MSRP6 | RP06 |
| 7 | .MSRP7 | RP07 |
| 11 | .MSRM3 | RMO3 |
| 24 | .MSR20 | RP20 |
| 27 | .MSR80 | RA80 |
| 30 | .MSR81 | RA81 |
| 31 | .MSR60 | RA60 |

B18(MS%SVD) Unit is online (in use) by another system through the software MSCP disk server.

B19(MS%IAC) Unit is temporarily inaccessible while the monitor checks the homeblocks to insure cluster integrity.

4 .MSRSN Byte pointer to ASCIZ string in which to store the structure name. This pointer is updated on return.

5 .MSRSA Byte pointer to ASCIZ string in which to store the structure alias. The alias is usually the same as the structure name. The alias is returned, and the pointer updated, only if the structure is on line.

6 .MSRNS Logical unit number within the structure of this unit in the left half, and number of units in the structure in the right half.

7 .MSRSW Number of pages for swapping on this structure.

10-12 .MSRUI Unit ID (3 words of 11-formatted ASCII)

13-15 .MSROI Owner ID (3 words of 11-formatted ASCII)

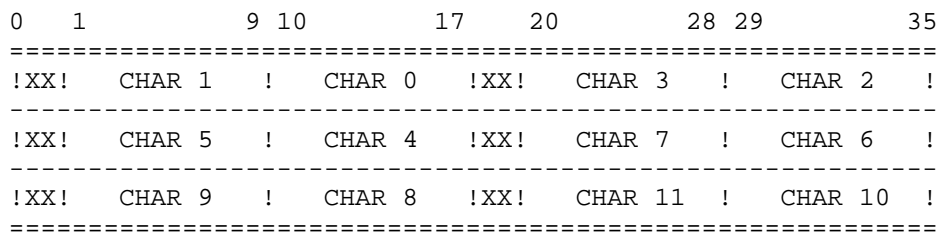
16-20 .MSRFI File system ID (3 words of 11-formatted ASCII)

TOPS-20 MONITOR CALLS
(MSTR)

| | | |
|----|--------|---|
| 21 | .MSRSP | Number of sectors per page |
| 22 | .MSRSC | Number of sectors per cylinder |
| 23 | .MSRPC | Number of pages per cylinder |
| 24 | .MSRCU | Number of cylinders per unit |
| 25 | .MSRSU | Number of sectors per unit |
| 26 | .MSRBT | Number of bit words in bit table per cylinder |
| 27 | .MSRSE | Serial number of the CPU for which the structure is used in booting the system |
| 30 | .MSRLS | Number of lost sectors per cylinder |
| 31 | .MSRSS | Number of sectors per surface |
| 32 | .MSDSH | High order serial number of disk drive |
| 33 | .MSDSN | Low order serial number of disk drive |
| 34 | .MSTSP | True number of sectors per page |
| 35 | .MSMID | Disk pack maintenance identifier. This number is the same for all packs in a structure. |

The length of the argument block in words is given by symbol .MSRLN.

The 11-formatted ASCII mentioned above is 7-bit ASCII stored four bytes to a 36-bit word in a format similar to that of a PDP-11:



The following errors are possible on the failure of this function.

- MSTRX2: WHEEL or OPERATOR capability required
- MSTRX3: Argument block too small
- MSTX14: Invalid channel number
- MSTX15: Invalid unit number
- MSTX16: Invalid controller number

TOPS-20 MONITOR CALLS
(MSTR)

MSTX18: No more units in system
MSTX27: Specified unit is not a disk
CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required

Obtaining the Status of a Given Disk Unit - .MSRUS

This function returns the status of the given disk unit. It accepts the channel, controller, and unit numbers in the first three words of the argument block. After successful completion of this function, the channel, controller, and unit numbers are unchanged, and the software information about the given disk unit is returned in the argument block.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

The difference between this function and the .MSRNU function is that .MSRUS does not search for the next disk unit but rather returns the status for the given unit. The .MSRNU function searches for the next disk unit and returns the status for that unit.

The format of the argument block is the same as described for the .MSRNU function.

Mounting a Given Structure - .MSMNT

This function brings the given structure on line and normally makes it available for general use. Any structure other than the public structure must be brought on line with this function. (The public structure is brought on line during the system startup procedure.)

.MSMNT can also be used to limit access to structures mounted on a system running the Common File System, CFS-20. Depending upon the setting of the exclusive bit, MS%EXL, structure can be mounted as sharable or exclusive. Sharable structures can be accessed by any job running on any processor on the CI, as long as that processor has not excluded the specified structure. Exclusive structures can only be accessed by jobs running on the processor that has the structure mounted.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

It is recommended that the .MSRNU (Read Next Unit) function be given first to locate all units in the structure. Then the .MSMNT (Mount Structure) function can be given to read and verify the HOME blocks of each unit and to mount the structure. If one or more units of the structure are write-locked, the structure cannot be mounted and an error is given.

The format of the argument block is as follows:

TOPS-20 MONITOR CALLS
(MSTR)

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .MSTNM | Pointer to the ASCIZ string containing the name of the structure (colon not allowed). |
| 1 | .MSTAL | Pointer to the ASCIZ string containing the alias of the structure. |
| 2 | .MSTFL | Flag bits in the left half, and the number of units in the structure (.MSTNU) in the right half. The bits that can be set in the left half are: <p style="margin-left: 2em;">B0(MS%NFH) If one of the HOME blocks is incorrect, do not fix it, but do return an error. If one of the HOME blocks is incorrect and this bit is off, the correct block is copied into the bad HOME block, and the mounting procedure continues.</p> <p style="margin-left: 2em;">B1(MS%NFB) If one of the BAT (Bad Allocation Table) blocks is incorrect, do not fix it and do return an error. If this bit is off and one of the BAT blocks is incorrect, the correct block is copied into the bad BAT block and the mounting procedure continues.</p> <p style="margin-left: 2em;">B2(MS%XCL) Mount the structure for exclusive use by this job. This bit is set by a system program when it initializes or reconstructs a structure. If this bit is off, the structure is mounted for general use.</p> <p style="margin-left: 2em;">B3(MS%IGN) Ignore correctable errors in the bit table and in the root directory on this structure. This bit is set by a system program when it reconstructs the root directory on a structure or rebuilds the bit table. If this bit is off and an error is detected, this function returns an error.</p> <p style="margin-left: 2em;">B4(MS%EXL) Mount structure exclusive to this processor. If this bit is set, only jobs running on the processor on which the structure is mounted may access files on that structure.</p> |
| 3 | .MSTUI | Beginning of unit information for each unit in the |

TOPS-20 MONITOR CALLS
(MSTR)

structure. The information is 3 words long per unit, and the symbol for this length is .MSTNO. The first 3-word block is for logical unit 0, and the last 3-word block is for the last logical unit (.MSTNU-1). The offsets into the 3-word block are:

| | | |
|---|--------|---|
| 0 | .MSTCH | Channel number of unit |
| 1 | .MSTCT | Controller number of unit (currently must be -1) |
| 2 | .MSTUN | Unit number of unit |

The number of argument words per unit is given by symbol .MSTNO (3).

After successful completion of this function, the given structure is mounted and available for general use (unless bit MS%XCL was on in word .MSTFL of the argument block). The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required
MSTRX3: Argument block too small
MSTRX4: Insufficient system resources
MSTRX5: Drive is not on line
MSTRX6: Home blocks are bad
MSTRX7: Invalid structure name
MSTRX8: Could not get OFN for ROOT-DIRECTORY
MSTRX9: Could not MAP ROOT-DIRECTORY
MSTX10: ROOT-DIRECTORY bad
MSTX11: Could not initialize Index Table
MSTX12: Could not OPEN Bit Table File
MSTX13: Backup copy of ROOT-DIRECTORY is bad
MSTX14: Invalid channel number
MSTX15: Invalid unit number
MSTX16: Invalid controller number
MSTX17: All units in a structure must be of the same type
MSTX19: Unit is already part of a mounted structure
MSTX20: Data error reading HOME blocks
MSTX23: Could not write HOME blocks
MSTX25: Invalid number of swapping pages
MSTX27: Specified unit is not a disk
MSTX30: Incorrect Bit Table counts on structure
MSTX34: Unit is write-locked
MSTX35: Too many units in structure
MSTX44: Mount type refused by another CFS processor
MSTX45: Structure naming or drive serial number conflict in CFS cluster
MSTX47: Shared access denied; already set exclusive in CFS cluster
MSTX48: Exclusive access denied; access conflict in CFS cluster

TOPS-20 MONITOR CALLS
(MSTR)

MSTX49: Structure naming conflict in CFS cluster
MSTX50: Mount type refused by this CFS processor
MSTX51: Insufficient system resources (structure limit exceeded)
MONX01: Insufficient system resources

Dismounting a Given Structure - .MSDIS

This function indicates that the given structure can be removed from the system. Any mounted structure other than the public structure (usually called PS:) can be dismounted with this function. (The public structure is dismounted at system shutdown.)

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

Files that are open at the time this function is executed become inaccessible, and the jobs that had the files open receive an error if they reference them. Jobs that have mounted the structure or have connected to or accessed a directory on the structure receive an informational message on the terminal. This message is

[STRUCTURE name: HAS BEEN DISMOUNTED]

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSDNM | Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure. |

After successful completion of this function, the given structure is dismounted and can be physically removed from the system.

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required
MSTRX3: Argument block too small
MSTX21: Structure is not mounted
MSTX24: Illegal to dismount the Public Structure

Obtaining the Status of a Given Structure - .MSGSS

This function returns the status of a mounted structure. It supplies the designators for the structure and for the storage of the structure's physical ID. After successful completion of the call, data is returned in the appropriate words in the argument block.

The format of the argument block, whose length is .MSGLN, is as follows:

TOPS-20 MONITOR CALLS
(MSTR)

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSGSN | Byte pointer to ASCIZ string containing the alias of the structure, or device designator of the structure. |
| 1 | .MSGST | Returned status word. The status bits are: <ul style="list-style-type: none"> B0(MS%PS) This structure is the login structure. B1(MS%DIS) This structure is being dismounted and no further mount count increments are allowed. B2(MS%DOM) This structure is a domestic structure. B3(MS%PPS) This structure is a permanent, protected structure. B4(MS%INI) This structure is being initialized. B5(MS%LIM) Directories on this structure are limited to the size of a directory on a DECSYSTEM-2050 (30 pages). B6(MS%NRS) Structure is non-regulated. B7(MS%RWS) Read-after-write checking is being done in the swapping area. B8(MS%RWD) Read-after-write checking is being done in the data area. B9(MS%ASG) Disk assignments are prohibited because bit table is bad. B10(MS%MXB) Bit table is too large for the monitor address space. B11(MS%CRY) Password encryption is enabled. B12(MS%IDT) Enable password invalidation by date. B13(MS%IVS) Enable password invalidation by use. B14(MS%DMP) Structure is dumpable. B15(MS%EXC) Structure is mounted exclusive to this processor; if off, the structure may be shared by other systems on the CI. |

TOPS-20 MONITOR CALLS
(MSTR)

| | | | |
|---|--------|-------------|---|
| | | B16(MS%IDX) | Index table file for OFNs has been set up. |
| | | B17(MS%CRD) | The root directory is being created on this structure. |
| | | B18(MS%OFS) | This structure is offline. |
| | | B19(MS%BS) | This structure is the Boot structure. |
| 2 | .MSGNU | | Number of units in structure. |
| 3 | .MSGMC | | Mount count for this structure. This value is determined by the number of .MSIMC (Increment Mount Count) functions given for this structure by all users since the structure was mounted. |
| 4 | .MSGFC | | Open file count (number of open files) for this structure. |
| 5 | .MSGSI | | Pointer to ASCIZ string in which to store the structure's physical ID. |

The length of the argument block is given by symbol .MSGLN (6).

After successful completion of this function, the status of the given structure is returned in the appropriate words of the argument block, and the pointer to the physical ID is updated to reflect the returned string.

The following errors are possible on the failure of this function.

MSTRX3: Argument block too small
MSTX21: Structure is not mounted

Changing the Status of a Given Structure - .MSSSS

This function changes the status of a mounted structure. The caller can change four of the status bits in the structure's status word: the status of being dismounted, the status of being domestic, the status of having read-after-write checking done in the swapping area of the disk, and the status of having read-after-write checking done in the data area.

RESTRICTIONS: Requires enabled WHEEL or OPERATOR capability.

The format of the argument block, the length of which is .MSSLN, is:

TOPS-20 MONITOR CALLS
(MSTR)

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSSSN | Byte pointer to ASCIZ string containing the alias of the structure, or device designator of the structure. |
| 1 | .MSSST | Word containing the new values for the bits being changed. |
| 2 | .MSSMW | Mask containing the bits being changed. The bits that can be changed are: <ul style="list-style-type: none"> B1(MS%DIS) Structure is being dismantled B2(MS%DOM) If set, structure is domestic; if not set, structure is foreign B6(MS%NRS) If set, structure is non-regulated; if not set, structure is regulated B7(MS%RWS) Read-after-write checking is being done in the swapping area B8(MS%RWD) Read-after-write checking is being done in the data area B14(MS%DMP) If set, structure is dumpable; if not set, structure cannot be dumped. |

The length of the argument block is given by symbol .MSSLN (3).

After successful completion of this function, the status of the given structure is changed according to the data supplied in the argument block.

The following errors are possible on the failure of this function.

- MSTRX2: WHEEL or OPERATOR capability required
- MSTRX3: Argument block too small
- MSTX21: Structure is not mounted
- MSTX22: Illegal to change specified bits

Initializing a Given Structure - .MSINI

This function creates a new structure or repairs an existing structure during normal system operation. The caller has the option of creating a new file system, reconstructing the root directory, writing a new set of HOME blocks on the structure, or rebuilding the index block.

RESTRICTIONS: Requires enabled WHEEL or OPERATOR capability.

TOPS-20 MONITOR CALLS
(MSTR)

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|------------|---|
| 0 | .MSINM | Byte pointer to ASCIZ string containing the name of the structure. |
| 1 | .MSIAL | Byte pointer to ASCIZ string containing the alias of the structure. |
| 2 | .MSIFL | Flag bits in B0-11, function value (MS%FCN) in B12-17, and number of units in structure (.MSINU) in B18-35. |
| | | Flag Bits |
| | B0(MS%NFH) | Do not fix HOME block if one is incorrect and do return an error. This bit can be on only with function .MSRRD. (See below.) |
| | B1(MS%NFB) | Do not fix BAT block if one is incorrect and do return an error. |
| | B2(MS%XCL) | Mount this structure for exclusive use by this job. If this bit is off, the structure is mounted for general use. |
| | B3(MS%IGN) | Ignore errors in the bit table and in the root directory on this structure. If this bit is on, B2(MS%XCL) must also be on. |
| | B4(MS%EXL) | Mount structure exclusive to this processor. If this bit is set, only jobs running on the processor on which the structure is mounted can access files on that structure. |
| | | Function Values |
| | 1 .MSCRE | Create a new file system |
| | 2 .MSRRD | Reconstruct the root directory |
| | 3 .MSWHB | Write a new set of HOME blocks |
| | 4 .MSRIX | Rebuild the index table |
| 3-5 | .MSISU | Beginning of unit information for each unit in the structure. The information is 3 words long per unit, and the symbol for this length is .MSINO. |

TOPS-20 MONITOR CALLS
(MSTR)

The first 3-word block is for logical unit 0, and the last 3-word block is for the last logical unit (.MSINU-1). The offsets into the 3-word block are:

- 0 .MSICH Channel number of unit
- 1 .MSICT Controller number of unit (currently must be -1)
- 2 .MSIUN Unit number of unit

The number of arguments per unit is given by symbol .MSINO (3).

- 6 .MSIST Status word (reserved for future use).
- 7 .MSISW Number of pages for swapping on this structure.
- 10 .MSIFE Number of pages for the front-end file system.
- 11-13 .MSIUI Unit ID (3 words of ASCII)
- 14-16 .MSIOI Owner ID (3 words of ASCII)
- 17-21 .MSIFI File system ID (3 words of ASCII) (reserved for future use)
- 22 .MSIFB Number of pages for the file BOOTSTRAP.BIN.
- 23 .MSISN Serial number of the CPU for which this structure is used in booting system. You must supply this word when creating a system structure that does not have the name PS:.

Words 6 through 23 (.MSIST through .MSISN) of the argument block must be supplied when the MSTR call is being executed to create a new file system or to write a new set of HOME blocks. After successful completion of the .MSCRE function, the structure is initialized and the following directories are created:

- <ROOT-DIRECTORY>
- <SYSTEM>
- <SUBSYS>
- <ACCOUNTS>
- <SPOOL>
- <OPERATOR>
- <SYSTEM-ERROR>

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required

TOPS-20 MONITOR CALLS
(MSTR)

MSTRX3: Argument block too small
MSTRX4: Insufficient system resources
MSTRX5: Drive is not on line
MSTRX6: Home blocks are bad
MSTRX7: Invalid structure name
MSTRX8: Could not get OFN for ROOT-DIRECTORY
MSTRX9: Could not MAP ROOT-DIRECTORY
MSTX10: ROOT-DIRECTORY bad
MSTX11: Could not initialize Index Table
MSTX12: Could not OPEN Bit Table File
MSTX13: Backup copy of ROOT-DIRECTORY is bad
MSTX14: Invalid channel number
MSTX15: Invalid unit number
MSTX16: Invalid controller number
MSTX17: All units in a structure must be of the same type
MSTX19: Unit is already part of a mounted structure
MSTX20: Data error reading HOME blocks
MSTX23: Could not write HOME blocks
MSTX25: Invalid number of swapping pages
MSTX26: Invalid number of Front-End-File system pages
MSTX27: Specified unit is not a disk
MSTX28: Could not initialize Bit Table for structure
MSTX29: Could not reconstruct ROOT-DIRECTORY
MSTX30: Incorrect Bit Table counts on structure
MSTX50: Mount type refused by this CFS processor
MSTX51: Insufficient system resources (structure limit exceeded)
MONX01: Insufficient system resources

Incrementing the Mount Count for the Job - .MSIMC

Users indicate that they are actively using a structure by incrementing the structure's mount count. A nonzero mount count informs the operator that the structure should not be dismounted. Also, an IPCF message is sent to the Mountable Device Allocator to indicate that a user is using the structure. The .MSIMC function is used to increment a structure's mount count.

Note that incrementing the mount count is a requirement for accessing files and directories on regulated structures.

The job receives an error if the given structure is in the process of being dismounted (a job has given the .MSSSS function with the MS%DIS bit on), or if the job is not logged in.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSDEV | Device designator, or byte pointer to ASCII string containing the alias of the structure. |

TOPS-20 MONITOR CALLS
(MSTR)

1 .MSJOB (Optional) Number of job (other than the current job) whose mount count is to be incremented. This requires WHEEL or OPERATOR capability to be enabled.

After successful completion of this function, the mount count of the given structure has been incremented.

The following errors are possible on the failure of this function.

ARGX18: Invalid structure name
CACTX2: Job is not logged in
LOUTX2: Invalid job number
MSTRX3: Argument block too small
MSTX21: Structure is not mounted
STRX10: Structure is offline
MSTX31: Structure already mounted
MSTX33: Structure is unavailable for mounting
MONX01: Insufficient system resources
STDVX1: No such device
STRX01: Structure is not mounted
STRX02: Insufficient system resources

Decrementing the Mount Count for the Job - .MSDMC

This function indicates that the given structure is no longer being used by the job executing the call. If the job executing the call has previously incremented the mount count for this structure via the .MSIMC (Increment Mount Count) function, the mount count is decremented. If the job has not incremented the mount count, the job receives an error. If the structure is regulated, and the user has any assigned JFNs on the structure, is accessing the structure or is connected to the structure, an error is returned.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSDEV | Device designator, or byte pointer to ASCIZ string containing the alias of the structure. |
| 1 | .MSJOB | (Optional) Number of job (other than the current job) whose mount count is to be decremented. This requires WHEEL or OPERATOR capability to be enabled. |

The resource allocator receives an IPCF packet when the mount count for a structure is decremented. The flag word (.IPCFL) of the packet descriptor block has a code of 1(.IPCCC) in the IP%CFC field (bits 30-32). This code indicates the message was sent by the monitor. The

TOPS-20 MONITOR CALLS
(MSTR)

first word of the packet data block contains the structure dismount code .IPCDS. The second word contains the number of header words and the number of the job decrementing the mount count. The third word contains the device designator of the structure. Thus,

.IPCFL/<.IPCC>B32

DATA/.IPCDS
DATA+1/number of header words (2),, job number
DATA+2/device designator of structure

After successful completion of this function, the mount count of the structure has been decremented and the IPCF message has been sent.

The following errors are possible on the failure of this function.

MSTRX3: Argument block too small
MSTX21: Structure is not mounted
MSTX32: Structure was not mounted
MSTX36: Illegal while JFNs assigned
MSTX37: Illegal while accessing or connected to a directory
ARGX18: Invalid structure name
MONX01: Insufficient system resources
STDVX1: No such device
STRX01: Structure is not mounted
STRX02: Insufficient system resources

Obtaining the Users on a Given Structure - .MSGSU

This function returns the job numbers of the users of the given structure. Users of a structure are divided into three classes: users who have incremented the mount count (MOUNT STRUCTURE command), users who are connected to the structure (CONNECT command), and users who have accessed the structure (ACCESS command). The caller specifies the classes of users for which information is to be returned by setting the appropriate bits in the argument block.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSUAL | Byte pointer to ASCIZ string containing the alias of the structure, or device designator of the structure. |
| 1 | .MSUFL | Flag bits in the left half and 0 in the right half. The bits that can be set are: B0(MS%GTA) Return users who have accessed the structure. |

TOPS-20 MONITOR CALLS
(MSTR)

B1(MS%GTM) Return users who have incremented the mount count.

B2(MS%GTC) Return users who are connected to the structure.

After successful execution of this function, word 1 through word n+1 (where n is the number of items returned) are updated with the following information.

| Word | Symbol | Meaning |
|-------|--------|---|
| 1 | .MSUFL | Right half contains the number of items (n) being returned. Left half is unchanged. |
| 2 | .MSUJ1 | Flag bits for the job in the left half, and number of job in the right half. |
| . | . | . |
| . | . | . |
| . | . | . |
| n + 1 | | Flag bits for the job in the left half, and number of job in the right half. |

The bits returned for each job are defined as:

B0(MS%GTA) Job has accessed structure.

B1(MS%GTM) Job has incremented the mount count for structure.

B2(MS%GTC) Job has connected to structure.

The following errors are possible on the failure of this function.

- MSTRX1: Invalid function
- MSTRX3: Argument block too small
- STRX01: Structure is not mounted
- STDVX1: No such device
- ARGX18: Invalid structure name
- MONX01: Insufficient system resources

Specifying Word and Bits To Be Modified - .MSHOM

This function allows an enabled WHEEL or OPERATOR program to modify a word of the homeblock of a mounted structure.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

TOPS-20 MONITOR CALLS
(MSTR)

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .MSHNM | Handle on alias such as pointer to string, or device designator. |
| 1 | .MSHOF | Offset specifying which word should be changed. |
| 2 | .MSHVL | Value for new bits. |
| 3 | .MSHMK | Mask showing which bits should be changed. |

The following errors are possible on the failure of this function:

MSTRX2: Insufficient privileges
MSTRX3: Argument block too small
MSTX21: Structure not mounted
STRX10: Structure is offline
Any errors "MODHOM" routine returns

Incrementing the Mount Count for the Fork - .MSICF

This function and the next (.MSDCF) allow job forks to independently mount and dismount structures without contending with one another for control of the structure. (This is primarily intended for SYSJOB.) Note that when either a job mount or fork mount is possible, the job mount is preferred as it incurs less overhead.

This function indicates that a fork is actively using a structure. If the structure is being dismounted, the job receives an error. The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSDEV | Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure. |

The following errors are possible on the failure of this function.

MSTRX3: Argument block too small
MSTX21: Structure is not mounted
MSTX33: Structure is unavailable for mounting
ARGX18: Invalid structure name
MONX01: Insufficient system resources
STDVX1: No such device
STRX01: Structure is not mounted
STRX02: Insufficient system resources

TOPS-20 MONITOR CALLS
(MSTR)

Decrementing the Mount Count for the Fork - .MSDCF

This function indicates that a fork is no longer using a structure. Note that if a job-wide increment has been done, the fork may still access the structure. The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSDEV | Pointer to ASCIZ string containing the alias of the structure, or device designator of the structure. |

The following errors are possible on the failure of this function.

MSTRX3: Argument block too small
MSTX21: Structure is not mounted
MSTX32: Structure was not mounted
MSTX36: Illegal while JFNs assigned
MSTX37: Illegal while accessing or connected to a directory
ARGX18: Invalid structure name
MONX01: Insufficient system resources
STDVX1: No such device
STRX01: Structure is not mounted
STRX02: Insufficient system resources

Receiving Interrupt when Disk Comes On-line - .MSOFL

This function specifies who is to receive an interrupt when a disk comes on-line. It is provided for the Mountable Device Allocator in order to control the disks and inform the operator of structure status. Only one process on the system will receive the interrupts.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

The argument block has the following format:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .MSCHN | Place this process on a software interrupt channel. An interrupt is then generated when a disk comes on-line. If the channel number is given as -1, a previously assigned interrupt channel will be deassigned. |

Ignoring Increment Check for Structure Use - .MSIIC

Allows a process to use a regulated structure without previously incrementing the mount count. Entries are made to the accounting file only on structure decrements, so this function will enable bypassing of accounting.

TOPS-20 MONITOR CALLS
(MSTR)

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

There is no argument block.

The following errors are possible on the failure of this function.

MSTRX2: WHEEL or OPERATOR capability required

Converting the Structure Mount Attribute - .MSCSM

This function may be used to change the mount attribute of a structure on a CFS-20 system. Under CFS-20, a structure may be mounted as sharable with other processors on the CI, or exclusive to a particular processor. Exclusive structures can only be accessed by jobs running on the owning processor.

The structure may be mounted with MSTR% function .MSMNT with the exclusive bit on or off. This function, .MSCSM, may be used to change the setting of the exclusive bit while the structure is mounted.

RESTRICTIONS: Requires enabled WHEEL or OPERATOR capability, and CFS-20 software.

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|------------|---|
| 0 | .MSCDV | Structure device designator |
| 1 | .MSCST | New mount attribute |
| | B4(MS%EXL) | 0 to set structure sharable 1 to set structure exclusive |

The following errors are possible on the failure of this function.

MSTRX1: Invalid function
MSTRX2: WHEEL or OPERATOR capability required
MSTRX3: Argument block too small
MSTX44: Mount type refused by another CFS processor
MSTX46: Illegal to specify mount attribute
MSTX47: Shared access denied; already set exclusive in CFS cluster
MSTX48: Exclusive access denied; access conflict in CFS cluster
MSTX50: Mount type refused by this CFS processor
MSTX51: Insufficient system resources (structure limit exceeded)
MONX02: Insufficient system resources (JSB full)
STRX01: Structure is not mounted

TOPS-20 MONITOR CALLS
(MTALN)

Associates a given serial-numbered magnetic tape drive with the specified logical unit number. The MTALN call is a temporary call and may not be defined in future releases.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Slave type in left half; logical unit number of magtape in right half

AC2: Decimal serial number of magnetic tape drive

RETURNS +1: Always

All units are searched for the specified serial number and slave type. When they are found, the drive is associated with the given logical unit number. The original unit is now associated with the logical unit number that the specified serial-numbered drive had before it was reassigned.

The slaves recognized are

| | |
|--------|---------------------------|
| .MTT45 | TU45 (The system default) |
| .MTT70 | TU70 |
| .MTT71 | TU71 |
| .MTT72 | TU72 |
| .MTT77 | TU77 |
| .MTT78 | TU78 |

Generates an illegal instruction interrupt on error conditions below.

MTALN ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required
DEVX1: Invalid device designator
OPNX7: Device already assigned to another job

Performs various device-dependent control functions. This monitor call requires either that the JFN be opened or the device be assigned to the caller if the device is an assignable device.

Because of the device dependencies of the MTOPR call, programs written

TOPS-20 MONITOR CALLS
(MTOPR)

with device-independent code should not use this call unless they first check for the type of device.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled. Some functions DECnet software.

ACCEPTS IN AC1: JFN of the device

AC2: Function code (see below)

AC3: Function arguments or address of argument block (see descriptions of individual devices)

RETURNS +1: Always

The functions listed for each device apply only to that device. If a function applies to more than one device, its description is repeated for each applicable device.

DECnet Functions

DECnet-20 MTOPR functions are described below. For a complete description of their application, see the DECnet manual.

| Code | Symbol | Meaning |
|------|--------|---|
| 24 | .MOACN | Allow a network task to enable software interrupt channels for any combination of the following work types: |

- o connect event pending
- o interrupt message available
- o data available

This function requires that AC3 contain three 9-bit fields specifying the changes in the interrupt assignments for this link. These fields are:

| Field | Symbol | Used to Signal |
|--------|--------|-----------------------------|
| B0-8 | MO%CDN | Connect event pending |
| B9-17 | MO%INA | Interrupt message available |
| B18-26 | MO%DAV | Data available |

The contents of the fields are

| Value | Meaning |
|-------|--|
| nnn | The number of the channel to be enabled; 0-5 and 23-35 decimal |

TOPS-20 MONITOR CALLS
(MTOPR)

.MOCIA Clear the interrupt
.MONCI No change

25 .MORLS Read the link status and return a 36-bit word of information regarding the status of the logical link. AC3 contains flag bits in the left half and a disconnect code in the right half. The flag bits are

| Symbol | Bit | Meaning |
|--------|-----|---|
| MO%CON | B0 | Link is connected |
| MO%SRV | B1 | Link is a server |
| MO%WFC | B2 | Link is waiting for a connection |
| MO%WCC | B3 | Link is waiting for a connection confirmation |
| MO%EOM | B4 | Link has an entire message to be read |
| MO%ABT | B5 | Link has been aborted |
| MO%SYN | B6 | Link has been closed normally |
| MO%INT | B7 | Link has an interrupt message available |
| MO%LWC | B8 | Link has been previously connected |

The disconnect/reject codes are as follows:

| Symbol | Value | Meaning |
|--------|-------|--|
| .DCX0 | 0 | Reject or disconnect by object |
| .DCX1 | 1 | Resource allocation failure |
| .DCX2 | 2 | Destination node does not exist |
| .DCX3 | 3 | Remote node shutting down |
| .DCX4 | 4 | Destination process does not exist |
| .DCX5 | 5 | Invalid process name field |
| .DCX6 | 6 | Object is busy |
| .DCX7 | 7 | Unspecified error |
| .DCX8 | 8. | Third party aborted link |
| .DCX9 | 9. | User abort (asynchronous disconnect) |
| .DCX10 | 10. | Invalid node name |
| .DCX11 | 11. | Local node shut down |
| .DCX21 | 21. | Connect initiate with illegal destination address |
| .DCX22 | 22. | Connect confirm with illegal destination address |
| .DCX23 | 23. | Connect initiate or connect confirm with zero source address |

TOPS-20 MONITOR CALLS
(MTOPR)

| | | |
|--------|-----|---|
| .DCX24 | 24. | Flow control violation |
| .DCX32 | 32. | Too many connections to node |
| .DCX33 | 33. | Too many connections to destination process |
| .DCX34 | 34. | Access not permitted |
| .DCX35 | 35. | Logical link services mismatch |
| .DCX36 | 36. | Invalid account |
| .DCX37 | 37. | Segment size too small |
| .DCX38 | 38. | No response from destination, process aborted |
| .DCX39 | 39. | No path to destination node |
| .DCX40 | 40. | Link aborted due to data loss |
| .DCX41 | 41. | Destination process does not exist |
| .DCX42 | 42. | Confirmation of disconnect initiate |
| .DCX43 | 43. | Image data field too long |

If a disconnect code does not apply to the current status of the link, the right half of AC3 will be zero.

26 .MORHN Return the ASCII name of the host node at the other end of the logical link. This function requires that AC3 contain a string pointer to the location where the host name is to be stored. (If the byte size exceeds eight bits, bytes are truncated to eight bits.)

The monitor call returns with an updated pointer in AC3, and the host name stored as specified.

This function is valid only for target tasks.

27 .MORTN Return the unique task name that is associated with your end of the logical link. If you had defaulted the task name in the network file specification, the call returns the monitor-supplied task name. In DECnet-20, the default task name is actually a unique number.

This function requires that AC3 contain a string pointer to the location where the task name is to be stored. (If the byte size exceeds eight bits, bytes are truncated to eight bits.)

The monitor call returns with an updated pointer in AC3 and the task name stored as specified.

30 .MORUS Return the source task user identification supplied in the connect initiate message. This

TOPS-20 MONITOR CALLS
(MTOPR)

function requires that AC3 contain a string pointer to the location where the user identification is to be stored. (If the byte size exceeds eight bits, bytes are truncated to eight bits.)

The monitor call returns with an updated pointer in AC3 and the user identification stored as specified. If no user identification was supplied by the source task, AC3 continues to point to the beginning of the string, and a null is returned as the only character.

- 31 .MORPW Return the source task's password as supplied in the connect initiate message. This function requires that AC3 contain a string pointer to the location where the password is to be stored. (Passwords are binary; therefore, the string pointer should accomodate 8-bit bytes.)

The monitor call returns with an updated pointer in AC3 and the source task's password stored as specified. AC4 contains the number of bytes in the string; a zero value indicates that no password was supplied by the source task.

- 32 .MORAC Returns the account string supplied by the source task in the connect initiate message. This function requires that AC3 contain a string pointer to the location where the account string is to be stored. (If the byte size exceeds eight bits, bytes are truncated to eight bits.)

The monitor call return with an updated pointer in AC3 and the source task's account number stored as specified. If no account string was supplied by the source task, AC3 continues to point to the beginning of the string, and a null is returned as the only character.

- 33 .MORDA Return the optional data supplied in any of the connect or disconnect messages. This function requires that AC3 contain a string pointer to the location where the optional user data is to be stored. (This file is binary; the string pointer should specify 8-bit bytes.)

The monitor call returns with an updated pointer in AC3 and the optional data stored as specified. AC4 contains the number of bytes in the data

TOPS-20 MONITOR CALLS
(MTOPR)

string; a zero value indicates that no optional data was supplied.

34 .MORCN Return the object type that was used by the source task to address this connection. The result indicates whether the local task was addressed by its generic type or its unique network task name.

The monitor call returns with the object type in AC3. A zero object type indicates that the target task was addressed by its unique network task name; a nonzero value indicates that it was addressed by its generic object type.

35 .MORIM Read interrupt message. This function requires that AC3 contain a byte pointer to the receiving buffer. (If the byte size exceeds eight bits, bytes are truncated to eight bits.) The maximum message length is 16 bytes, and the buffer size should be at least 8 bits.

The monitor call returns with an updated pointer in AC3, the message stored in the buffer, and the count of bytes received in AC4.

36 .MOSIM Send an interrupt message. This function requires that AC3 contain a byte pointer to the message (8-bit maximum) and that AC4 contain a count of the bytes in the interrupt message (16-byte maximum).

40 .MOCLZ Reject a connection either implicitly or explicitly. If the target task closes its JFN (via the CLOSF monitor call) before accepting the connection either implicitly or explicitly, the local NSP assumes that the connection is rejected and sends a connect reject message back to the source task. The reason given is process aborted (reject code 38, .DCX38). The target task must then reopen its JFN in order to receive subsequent connect initiate messages. In order to explicitly reject a connect and at the same time return a specific reject reason and set up 16 bytes of user data, the target task must use the .MOCLZ function of the MTOPR monitor call. The .MOLCZ function does not close the JFN.

The function requires the following:

1. AC2 contain a reject code in the left half and .MOCLZ in the right half. The

TOPS-20 MONITOR CALLS
(MTOPR)

reject code is a 2-byte, NSP-defined decimal number indicating the reason that a target task is rejecting a connection. See the description of code 25, .MORLS, for a list of disconnect/reject codes.

2. AC3 contain a string pointer to any data to be returned. (If the byte size exceeds eight bits, bytes are truncated to eight bits.)
3. AC4 contain the count of bytes in the data string (maximum=16). A zero indicates no data.

41 .MOCC Accept a connection explicitly. Under certain conditions, the local NSP assumes that the connection is accepted and sends a connect confirm message back to the source task. These implicit conditions are the following:

1. The target task attempts to output to the logical link (issues a SOUT or SOUTR monitor call to the network).
2. The target task submits a read request to the logical link (issues a SIN or SINR monitor call to the network).

In order to explicitly accept a connect and also return a limited amount of data, the target task must use the .MOCC function of the MTOPR monitor call. This function requires that AC3 contain a string pointer to any data to be returned. (If byte size exceeds eight bits, bytes are truncated to eight bits.) AC4 must contain the count of bytes in the data string to a maximum of 16 bytes. A zero indicates no data.

42 .MORSS Returns the maximum segment size that can be sent over this link. This value is the minimum of the maximum segment size supported by the remote NSP task, the segment size supported by the remote network task, and the segment size supported by the local NSP task. The local task can use this value to optimize the format of data being transmitted over the link. This function is illegal if the link is not in run state.

The monitor call returns the maximum segment size, in bytes, in AC3.

TOPS-20 MONITOR CALLS
(MTOPR)

44 .MOSNH Sets the network host. This function causes the terminal specified in the argument block to send data to and receive data from the DECnet logical link. The link connects the terminal on the local host to a job on a foreign host. The DECnet logical link to the foreign host must be established by the user process before this MTOPR function can be executed.

This function requires the JFN of the logical link in AC1, and the address of the argument block in AC3. The argument block has the following format:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | | The length of the argument block including this word. |
| 1 | .SHTTY | Identifier of the terminal that is controlling the local job. |
| 2 | .SHESC | Flags in the left half, ASCII escape character in the right half. The flags defined are: |
| | | SH%LPM local page mode |

45 .MOSLP Set link parameters. This function causes the link parameters specified in the argument block to be set.

The process must have WHEEL or OPERATOR capability enabled to use this function.

This function must be called before the link is established (before the OPENF call for an active link, or before the MTOPR call that accepts a link for a passive link).

This function requires the address of the argument block be in AC3. The argument block has the following format:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | | The length of the argument block, including this word. |
| 1 | .SLPSS | The link segment size. The value actually used is the lowest of these 3 values: the segment size specified, the local node's maximum segment size, and the remote node's segment size. |

TOPS-20 MONITOR CALLS
(MTOPR)

2 .SLPFC The flow control option. The argument consists of two fields:

B15-B17 MO%RFC Remote end flow control
B33-B35 MO%LFC Local end flow control

If a value for the remote end flow control is given, it is ignored. The possible values for the local end flow control are:

| Value | Symbol | Meaning |
|-------|--------|----------------------|
| 1 | NSF.CO | No flow control |
| 2 | NSF.CS | Segment flow control |
| 3 | NSF.CM | Message flow control |

46 .MORLP Read link parameters. This function returns the link parameters. The arguments to this function are the same as those to .MOSLP (set link parameters) function.

No capabilities are required for this function. Returned value of -1 means that the parameters for the link have not yet been decided.

Note that the .MORSS MTOPR function can be used to retrieve the segment size. There is no difference between the value of segment size returned by the .MORSS function and the .MORLP function, once the link is established.

47 .MOSLQ Set link quotas. This function sets the parameters related to link quotas.

This function requires the address of an argument block in AC3. The argument block has the following format:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | | Length of the argument block, including this word. |
| 1 | .SLQIP | Percent of link quota used for input. However, a minimum of one buffer is reserved for input and output. |

TOPS-20 MONITOR CALLS
(MTOPR)

- 2 .SLQLQ Link quota. This function sets the quota of buffers for this logical link. The number of buffers used depends on the job quota, and on the availability of buffers. If the process does not have WHEEL or OPERATOR capability enabled, the default value is used instead.
- 3 .SLQIG Input goal. This function sets the goal for the number of outstanding input data requests. If the process does not have WHEEL or OPERATOR capability enabled, the default value is used instead.
- 50 .MORLQ Read link quota. The arguments to this function are the same as those to the .MOSLQ (set link quota parameters) function, and the values are returned in the argument block.
- 51 .MORFT Return the format type of the source process name.

The monitor call returns the format type in AC3. The following format types are defined:

| Value | Symbol | Meaning |
|-------|--------|---|
| 0 | .FMTT0 | Type 0. The user has specified a nonzero object type; the other fields must be zero or have a zero length. |
| 1 | .FMTT1 | Type 1. The user has not specified an object type; the PBOBJ field is zero. The user supplied a process name up to 16 bytes long in the PBNAM field. |
| 2 | .FMTT2 | Type 2. The user has not specified an object type; the PBOBJ field is zero. The monitor has filled in the PBGRP and PBUID fields with the ford number and job number, respectively. The monitor supplies the user's LOGINID up to 12 bytes long in the PBNAM field. |

TOPS-20 MONITOR CALLS
(MTOPR)

Front-End Functions

| Code | Symbol | Meaning |
|------|--------|---|
| 3 | .MOEOF | <p>Causes TOPS-20 to flush its buffers and send all data to the front end. Optionally, it will notify the front end of the end-of-file condition. If AC3 is zero, the buffers are flushed and the end of file status is sent to the front end. If AC3 is nonzero, only the buffers are flushed.</p> <p>This function is used for synchronization between a program running on TOPS-20 and a program running on the front end.</p> |
| 4 | .MODTE | <p>Assign the specified device to the DTE controller on the front end. This function, which must be performed before I/O is allowed to the device, requires AC3 to contain the device type. The process must have WHEEL or OPERATOR capability enabled.</p> <p>Unless otherwise noted, the JFN must be opened before the MTOPR function can be performed.</p> |

MTA/MT Functions

The functions available for physical magnetic tape drives (MTA) and logical magnetic tape drives (MT) are described below. Some of these functions accept arguments in AC3 (see the individual descriptions). In the following descriptions, a labeled tape is one acquired via a MOUNT command and has one of the following attributes: ANSI, TOPS20, or EBCDIC.

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .MOCLE | Clear any error flags from a previous MTOPR call. |
| 1 | .MOREW | <p>Rewind the tape. This function waits for activity to stop before winding the tape. If sequential data is being output, the last partial buffer is written before the tape is rewound. Control returns to caller when rewinding begins. For labeled tapes, this function causes the first volume in the set to be mounted and positioned to the first file in the file set. Since a volume switch may be required, this function could block for a considerable amount of time.</p> <p>Use function .MORVL to rewind the current volume.</p> |

TOPS-20 MONITOR CALLS
(MTOPR)

2 .MOSDR Set the direction of the tape motions for read operations. This function requires AC3 to contain the desired direction. If AC3=0, the tape motion is forwards; if AC3=1, the tape motion is backwards.

This function is not available for labeled tapes and will return an MTOX1 error if used for that purpose.

3 .MOEOF Write a tape mark. This function requires that the magnetic tape be opened for write access. If sequential data is being output, the last partial buffer is written before the tape mark.

For labeled tapes, issuing this function will terminate the data portion of the file, write EOF trailer labels and leave the tape positioned to accept user trailer labels. It is possible at this point to write user trailer labels or close the file. A second .MOEOF function issued without positioning the tape backwards will "close" the file (subsequent writes will create a new file).

4 .MOSDM Set the hardware data mode to be used when transferring data to and from the tape. This function requires AC3 to contain the desired data mode:

- 0 .SJDDM default system data mode
- 1 .SJDMC dump mode (36-bit bytes)
- 2 .SJDm6 SIXBIT byte mode for 7-track drives
- 3 .SJDMA ANSI ASCII mode (7 bits in 8-bit bytes)
- 4 .SJDm8 industry compatible mode
- 5 .SJDmH High-density mode for TU70 and TU72 tape drives only (nine 8-bit bytes in two words).

For labeled tapes, this function is allowed only if the file is opened in dump mode (.GSDMP). If this is not the case, an MTOX1 error is returned.

5 .MOSRS Set the size of the records. This function requires AC3 to contain the desired number of bytes in the records. This function is allowed only if no I/O has been done since the JFN was opened.

This function is illegal for labeled tapes; an MTOX1 error is returned.

TOPS-20 MONITOR CALLS
(MTOPR)

The maximum size of the records (in bytes) is as follows:

| Hardware I/O Mode | Maximum Record Size (bytes) |
|---------------------------------|--------------------------------|
| System-default | --- |
| Dump (dump is usual default) | 8192 |
| SIXBIT | 49152 |
| ANSI ASCII | 40960 |
| Industry compatible | 32768 |
| High density | 8192 |

The above values can be exceeded in the execution of .MOSRS; however, the first data transfer will fail.

- 6 .MOFWR Advance over one record in the direction away from the beginning of the tape. If sequential data is being read in the forward direction and not all of the record has been read, this function advances to the start of the next record. If sequential data is being read in the reverse direction and not all of the record has been read, this function positions the tape at the end of that record.

For labeled tapes, forward space will position over a logical record. This implies that many physical records may be skipped (if S format is used) perhaps involving one or more volume switches.

- 7 .MOBKR Space backward over one record in the direction toward the beginning of the tape. If sequential data is being read in the forward direction and not all of the record has been read, this function positions the tape back to the start of that record. If sequential data is being read in the reverse direction and not all of the record has been read, this function positions the tape to the end of the record physically preceding that record.

For labeled tapes, backward spacing will position over a logical record. This implies that many physical records may be skipped (if S format is used) perhaps involving one or more volume switches.

- 10 .MOEOT For unlabeled tapes, advance forward until two

TOPS-20 MONITOR CALLS
(MTOPR)

sequential tape marks are seen and position tape after the first tape mark.

For labeled tapes, this function will position the volume set beyond the end of the last file in the set. This is useful for adding a new file to the end of an already existing volume set. This function may take some time to complete as one or more volumes switches may be required.

11 .MORUL Rewind and unload the tape. This function is identical to the .MOREW function and also unloads the tape if the hardware supports tape unloading.

This function is illegal for any tape acquired via the MOUNT command.

12 .MORDN Return the current density setting. On a successful return, AC3 contains the current density.

13 .MOERS Erase three inches of tape (erase gap). This function requires that the magnetic tape be opened for write access.

This function is illegal for labeled tapes.

14 .MORDM Return the hardware data mode currently being used in transfers to and from the tape. On a successful return, AC3 contains the current data mode.

15 .MORRS Return the size of the records. On a successful return, AC3 contains the number of bytes in the records.

16 .MOFWF Advance to the start of the next file. This function advances the tape in the direction away from the beginning of the tape until it passes over a tape mark.

For labeled tapes, forward space will skip one logical file. This implies that many physical files may be skipped, involving perhaps one or more volume switches.

17 .MOBKF Space backward over one file. This function moves the tape in the direction toward the beginning of the tape until it passes over a tape mark or reaches the beginning of the tape, whichever occurs first.

TOPS-20 MONITOR CALLS
(MTOPR)

For labeled tapes, backspace file will back up one logical file. This implies that many physical files may be skipped, involving perhaps one or more volume switches.

NOTE

For labeled ANSI tapes, the monitor can compute the number of volume switches required to get to the first section of the file. Thus, if this function is issued for an ANSI tape, at most one volume switch will be required. This is not true for EBCDIC tapes.

Issuing this function when the tape is already positioned at the first volume of the volume set will not produce an error. The program issuing this function must follow the .MOBKF with a GDSTS call to determine if the BOT was encountered during the backspacing operation.

- | | | |
|----|--------|--|
| 20 | .MOSPR | Set the parity. This function requires AC3 to contain the desired parity: 0 .SJPRO odd parity 1 .SJPRE even parity |
| 21 | .MORPR | Return the current parity. On a successful return, AC3 contains the current parity. |
| 22 | .MONRB | Return number of bytes remaining in the current record. On a successful return, AC3 contains the number of bytes remaining. This function is only meaningful during sequential I/O. |
| 23 | .MOFOU | Force any partial records to be written during sequential output. |
| 24 | .MOSDN | Set the density. The function requires AC3 to contain the desired density. 0 .SJDDN default system density 1 .SJDN2 200 BPI (8 rows/mm) 2 .SJDN5 556 BPI (22 rows/mm) 3 .SJDN8 800 BPI (31 rows/mm) 4 .SJD16 1600 BPI (63 rows/mm) 5 .SJD62 6250 BPI (246 rows/mm) |

This function is illegal for labeled tapes.

TOPS-20 MONITOR CALLS
(MTOPR)

25 .MOINF Return information about the tape. This function requires AC3 to contain the address of the argument block in which the information is to be returned. The format of the argument block is as follows:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .MOICT | Length of argument block to be returned (not including this word) |
| 1 | .MOITP | MTA type code |
| 2 | .MOIID | MTA reel ID |
| 3 | .MOISN | Channel, controller, and unit in the left half and serial number in the right half. |
| 4 | .MOIRD | Number of reads done |
| 5 | .MOIWT | Number of writes done |
| 6 | .MOIRC | Record number from beginning of tape |
| 7 | .MOIFC | Number of files on tape |
| 10 | .MOISR | Number of soft read errors |
| 11 | .MOISW | Number of soft write errors |
| 12 | .MOIHR | Number of hard read errors |
| 13 | .MOIHW | Number of hard write errors |
| 14 | .MOIRF | Number of frames read |
| 15 | .MOIWF | Number of frames written |
| 16 | .MOICH | Channel number |
| 17 | .MOICO | Controller number |
| 20 | .MOIUN | Unit number |
| 21 | .MOIDH | High order serial number of drive |
| 22 | .MOIDN | Low order serial number of drive |

The JFN need not be open for this function.

26 .MORDR Return the direction that the tape is moving during read operations. On a successful return, AC3=0 if the direction of the tape motion is forwards, or AC3=1 if the direction of the tape motion is backwards.

27 .MOSID Set the reel identification of the tape mounted. The process must have WHEEL or OPERATOR capability enabled. This function requires AC3 to contain the desired 36-bit reel ID. The JFN need not be open for this function.

30 .MOIEL Inhibit error logging for the tape. If AC3 is nonzero, error logging will be inhibited on subsequent operations on the tape drive. If AC3 is zero, error logging will be performed. The setting remains in effect until the JFN is closed.

TOPS-20 MONITOR CALLS
(MTOPR)

Error logging occurs by default if no setting is made with function .MOIEL.

- 31 .MONOP Wait for all activity to stop.
- 32 .MOLOC Specifies the first volume in a MOUNT request, or identifies the "next" volume for a volume switch. This function requires OPERATOR or WHEEL capability.

AC3 contains a pointer to an argument block having the following format:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .MOCNT | count of words in the block |
| 1 | .MOMTN | MT unit number to associate with this MTA |
| 2 | .MOLBT | label type (.LTxxx) |
| 3 | .MODNS | density |
| 4 | .MOAVL | address of volume labels |
| 5 | .MONVL | number of volume labels at .MOAVL |
| 6 | .MOCVN | volume number in the volume set |
| 7 | .MOVSN | SIXBIT file set identifier |

The JFN need not be open for this function.

- 37 .MOSTA Return current magtape status. Argument block has the following form and contents:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .MOCNT | Count of words in the block including this word (user-supplied) |
| 1 | .MODDN | Density flags (returned) |
| | | Bit Symbol Meaning |
| | | B1 SJ%CP2 200 BPI |
| | | B2 SJ%CP5 556 BPI |
| | | B3 SJ%CP8 800 BPI |
| | | B4 SJ%C16 1600 BPI |
| | | B5 SJ%C62 6250 BPI |
| 2 | .MODDM | Data mode flags (returned) |
| | | Bit Symbol Meaning |
| | | B1 SJ%CMC core dump |
| | | B2 SJ%CM6 SIXBIT |

TOPS-20 MONITOR CALLS
(MTOPR)

B3 SJ%CMA ANSI ASCII
 B4 SJ%CM8 industry compatible
 B5 SJ%CMH high density mode

3 .MOTRK Recording track flags (returned)

Bit Symbol Meaning

B1 SJ%7TR 7-track drive
 B2 SJ%9TR 9-track drive

4 .MOCST Tape status flags (returned)

Bit Symbol Meaning

B0 SJ%OFS off line
 B1 SJ%MAI maintenance mode enabled
 B2 SJ%MRQ maintenance mode requested
 B3 SJ%BOT beginning of tape
 B4 SJ%REW rewinding
 B5 SJ%WLK write locked

5 .MODVT Device type (returned)

Code Symbol Meaning

3 .MTT45 TU45 (system default)
 17 .MTT70 TU70
 20 .MTT71 TU71
 21 .MTT72 TU72
 13 .MTT77 TU77
 19 .MTT78 TU78

The JFN need not be open for this function.

40 .MOOFL Enable interrupts for online/offline transition. Allows a process to be interrupted if a magnetic tape drive's state changes from online to offline or vice-versa and when a rewind operation completes. This function must be performed once for each drive for which interrupts are to be enabled. If multiple drives are enabled for interrupts, then a .MOSTA function should be performed (for each drive) before interrupts for the drives are enabled. Then, when an interrupt occurs, .MOSTA can be performed for each drive and the current status of that drive can be compared against the previous status. Thus, it can be determined which drive (or drives) interrupted.

TOPS-20 MONITOR CALLS
(MTOPR)

This function requires OPERATOR or WHEEL capability. The JFN need not be open for this function.

42 .MOPST Declares the software interrupt channel to be used by the monitor to indicate that the UTL labels at the end-of-volume or the UHL labels at the start of the new volume are available. If this MTOPR is not performed before an EOVS label set is encountered, the user program will not be given the opportunity to process the UTL or UHL labels during the volume switch operation.

AC3 contains the PSI channel number to set. The channel can be cleared by using -1 in AC3.

This function is for labeled tapes only.

43 .MORVL Rewind current labeled tape volume. This function is for labeled tapes only.

44 .MOVLS Switch volumes for an unlabeled multi-volume set. If an unlabeled tape is mounted specifying multiple volumes in the volume set, the monitor will not automatically perform a volume switch at the end of each volume. The .MOVLS function may be issued in such a case to perform a volume switch. This function is legal only for unlabeled MT devices.

AC3 contains the address of an argument block having the following format:

Word Contents

| | |
|---|---|
| 0 | count of words in block including this word |
| 1 | flags,,function code |
| 2 | argument (if required) |

Available functions are:

| Word | Symbol | Function |
|------|--------|---|
| 1 | .VSMNV | mount absolute volume number (volume number in word 2 of the argument block) |
| 2 | .VSFST | mount first volume in set |
| 3 | .VSLST | mount last volume in set |
| 4 | .VSMRV | mount relative volume number (volume number in word 2 of the argument block). For |

TOPS-20 MONITOR CALLS
(MTOPR)

.VSMRV, the argument in word 2 of the argument block is the volume number relative to the current mounted volume to mount. For example, if volume #2 is currently mounted and .VSMRV is performed with 2 in word 2 of the argument block, then volume 4 will be mounted. Specifying 1 in word 2 of the argument block will mount the next volume in the set.

5 .VSFLS force volume switch for labeled tape. This function is only for tapes for which .MOSDS has previously been set.

45 .MONTR Set no translate.

Sets or clears the EBCDIC to ASCII translate flag. If the flag is set and the tape file being read is from an IBM EBCDIC volume, then all data delivered to the user program will be in its original EBCDIC form. If the flag is not set, and the file is from an IBM EBCDIC volume, then all data delivered to the user program will be in ASCII. In order to perform this translation, certain information may be lost (as the EBCDIC character set contains 256 codes while the ASCII character set contains only 128 codes - see Appendix A for ASCII-to-EBCDIC conversions). Note that the setting of this flag has no effect on the data delivered by the MTU% JSYS. This setting applies until explicitly changed or until the MT is dismounted. The default value of the flag is "clear" (translate).

If AC3 is zero, the translate flag is cleared. If AC3 is negative, the translate flag is set.

This function is for labeled tapes only. The JFN need not be open for this function.

46 .MORDL Read user header labels. Labels must be read immediately after the file is opened (and before the first input is requested) or after a volume switch has occurred and the volume switch PSI has been generated. .MORDL may be used to read either the UHL or UTL labels. User header labels may be read only if the file is opened for read or

TOPS-20 MONITOR CALLS
(MTOPR)

append. The labels may be a maximum of 76 characters long.

User trailer labels may be read at any time. If the program requests to read user trailer labels, the tape will be positioned to the EOF trailer section.

AC3 contains a byte pointer to the area for receiving the label.

On a successful return, AC2 contains the user label identifier. This will be the ASCII character following the UHL or the UTL. AC3 will contain an updated byte pointer.

This function is for labeled tapes only.

47 .MOWUL Write user header labels or user trailer labels. User header labels may be written only after the file is opened (and before the first write is performed) or when a PSI is generated, indicating that a volume switch has occurred. User header labels may be written only if the file is opened for write access.

User trailer labels may be read or written at any time. If the program requests to write user trailer labels, the file will be terminated with an EOF trailer section. Once user trailer labels are written in this manner, no more data may be read or written.

User trailer labels may also be written during a volume switch sequence. Once the PSI indicating EOVS has been received, the user program may write a UTL label into the EOVS trailer section. This operation must be performed at interrupt level.

AC3 contains a byte pointer to the label contents. This string must contain 76 bytes of data (the monitor will use only the first 76 bytes). AC4 contains a label identifier code (any ASCII character).

It is possible to encounter EOT while writing the first UTL in the EOF trailer set. This can occur if the last data write overwrote the EOT mark. In this instance, the user program will receive the EOVS PSI from within the code writing the UTL labels for the file. It is not possible to

TOPS-20 MONITOR CALLS
(MTOPR)

receive an EOVS while writing the trailer labels in the EOVS set.

This function is for labeled tapes only.

50 .MORLI Reads the available fields from the standard volume and header labels.

AC3 contains a pointer to an argument block of the form:

| Word | Contents | | | | | | | | | | | | | | | |
|-------|--|------------|--------|------------|---|--------|-----------|---|--------|------|---|--------|--------|---|--------|---------|
| 0 | count of words in block | | | | | | | | | | | | | | | |
| 1 | word to store label type of this tape | | | | | | | | | | | | | | | |
| | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Label Type</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>.LTUNL</td> <td>Unlabeled</td> </tr> <tr> <td>2</td> <td>.LTANS</td> <td>ANSI</td> </tr> <tr> <td>3</td> <td>.LTEBC</td> <td>EBCDIC</td> </tr> <tr> <td>4</td> <td>.LTT20</td> <td>TOPS-20</td> </tr> </tbody> </table> | Value | Symbol | Label Type | 1 | .LTUNL | Unlabeled | 2 | .LTANS | ANSI | 3 | .LTEBC | EBCDIC | 4 | .LTT20 | TOPS-20 |
| Value | Symbol | Label Type | | | | | | | | | | | | | | |
| 1 | .LTUNL | Unlabeled | | | | | | | | | | | | | | |
| 2 | .LTANS | ANSI | | | | | | | | | | | | | | |
| 3 | .LTEBC | EBCDIC | | | | | | | | | | | | | | |
| 4 | .LTT20 | TOPS-20 | | | | | | | | | | | | | | |
| 2 | byte pointer to area for storing volume name string | | | | | | | | | | | | | | | |
| 3 | byte pointer to area for storing owner name string | | | | | | | | | | | | | | | |
| 4 | word to store tape format (ASCII character) | | | | | | | | | | | | | | | |
| 5 | word to store record length | | | | | | | | | | | | | | | |
| 6 | word to store block length | | | | | | | | | | | | | | | |
| 7 | word to store creation date (in internal format) | | | | | | | | | | | | | | | |
| 10 | word to store expiration date (in internal format). Returns a -1 in this word if the date is invalid. | | | | | | | | | | | | | | | |
| 11 | byte pointer to area for storing file name string | | | | | | | | | | | | | | | |
| 12 | word to store generation number | | | | | | | | | | | | | | | |
| 13 | word to store version number | | | | | | | | | | | | | | | |
| 14 | word to store mode value (form-control value). The possible modes are as follows: | | | | | | | | | | | | | | | |

| Mode | |
|-------|---|
| Value | Meaning |
| space | no line format characters are present |
| A | FORTRAN format control characters are present |

TOPS-20 MONITOR CALLS
(MTOPR)

| | |
|---|--|
| M | All necessary line format characters are present |
| X | Data in stream mode |

The user specifies only the block count and the byte pointers; the remaining values are returned by the monitor. If a zero is substituted for any of the byte pointers, then the associated string is not returned.

This function is normally issued when the JFN is open. If issued when the JFN is closed, only the first 3 words of the argument block are returned. If the tape is unlabeled, only the first word of the argument block is returned. For labeled tapes only.

51 .MOSMV Declares the value to be placed in the DEC-defined "form-control" field in the HDR2 label. This field is not defined in the ANSI standard but should be specified whenever the data file is meant to be read with DEC-supplied software. This function merely declares the value to be placed in the label. It is the user program's responsibility to produce records that conform to the declared mode.

AC3 contains one of the following modes:

| Value | Symbol | Mode |
|-------|--------|--------------------------------------|
| 0 | .TPFST | X - (stream mode) |
| 1 | .TPFCP | M - (all formatting control present) |
| 2 | .TPFFC | A - (FORTRAN control present) |
| 3 | .TPFNC | space - (no controls present) |

This function is for labeled tapes only.

52 .MOSDS Set deferred volume switch. Inhibits the monitor from doing an automatic volume switch and allows a program to write its own trailer information beyond the physical end-of-tape mark. This function is intended for labeled MT devices open for writing in DUMP mode.

53 .MOIRB Return the block status of the DUMP mode operation. A 0 is returned if the request will not block, and a nonzero is returned if the request will block.

TOPS-20 MONITOR CALLS
(MTOPR)

PLPT Functions

The functions available for physical line printers (PLPT) are described below. Some of these functions accept the address of an argument block in AC3. The first word of the argument block contains the length (including this word) of the block. Remaining words of the block contain arguments for the particular function.

| Code | Symbol | Meaning | | | | | | | | |
|------|---|--|---|--------------------------------|---|------------------------------------|---|---------------------------------------|--|---|
| 27 | .MOPSI | <p>Enable for a software interrupt on nonfatal device conditions. Examples of these conditions are:</p> <ol style="list-style-type: none">1. Device changed from offline to online.2. Device changed from online to offline.3. Device's page counter has overflowed. <p>Other device errors or software conditions are not handled by this function; instead they cause a software interrupt on channel 11 (.ICDAE).</p> <p>Argument Block:</p> <p>Word Contents</p> <table border="0"><tr><td>0</td><td>word count including this word</td></tr><tr><td>1</td><td>interrupt channel number</td></tr><tr><td>2</td><td>flags. The following flag is defined:</td></tr><tr><td></td><td>B0(MO%MSG) Suppress standard CTY device messages.</td></tr></table> | 0 | word count including this word | 1 | interrupt channel number | 2 | flags. The following flag is defined: | | B0(MO%MSG) Suppress standard CTY device messages. |
| 0 | word count including this word | | | | | | | | | |
| 1 | interrupt channel number | | | | | | | | | |
| 2 | flags. The following flag is defined: | | | | | | | | | |
| | B0(MO%MSG) Suppress standard CTY device messages. | | | | | | | | | |
| 31 | .MONOP | <p>Wait for all activity to stop. This function blocks the process until all data has actually been sent to the printer and has been printed. Because this function is transferring data, it can return an IOX5 data error.</p> | | | | | | | | |
| 32 | .MOLVF | <p>Load the line printer's VFU (Vertical Formatting Unit) from the file indicated in the argument block.</p> <p>Argument Block:</p> <p>Word Contents</p> <table border="0"><tr><td>0</td><td>word count including this word</td></tr><tr><td>1</td><td>JFN of the file containing the VFU</td></tr></table> | 0 | word count including this word | 1 | JFN of the file containing the VFU | | | | |
| 0 | word count including this word | | | | | | | | | |
| 1 | JFN of the file containing the VFU | | | | | | | | | |

TOPS-20 MONITOR CALLS
(MTOPR)

The system opens the file for input with a byte size of 18 bits. It closes the file and releases the JFN when the loading of the VFU is complete.

33 .MORVF Read the name of the current VFU file stored in the monitor's data base.

Argument Block:

Word Contents

- 0 word count including this word
- 1 pointer to destination area for ASCIZ name string
- 2 number of bytes in destination area

34 .MOLTR Load the line printer's translation RAM (Random Access Memory) from the file indicated in the argument block.

Argument Block:

Word Contents

- 0 word count including this word
- 1 JFN of the file containing the translation RAM

The system opens the file for input with a byte size of 18 bits. It closes the file and releases the JFN when the loading of the translation RAM is complete.

35 .MORTR Read the name of the current translation RAM file stored in the monitor's data base.

Argument Block:

Word Contents

- 0 word count including this word
- 1 pointer to destination area for ASCIZ name string
- 2 number of bytes in destination area

36 .MOSTS Set the status of the line printer.

Argument Block:

TOPS-20 MONITOR CALLS
(MTOPR)

Word Contents

0 word count including this word
1 software status word, with the following status bits settable by the caller:

B0(MO%LCP) Set line printer as a lowercase printer.

B12(MO%EOF) Set bit MO%EOF in the printer status word when all data sent to printer has actually been printed. The status word can be obtained with the .MORST function.

B14(MO%SER) Clear the software error condition on the line printer. This condition usually occurs on a character interrupt.

Other status bits can be read with the .MORST function (see below) but cannot be set by the caller.

2 value for page counter register. The caller can indicate the number of pages to be printed by specifying a value of up to 12 bits (4096). Each time the printer reaches the top of a new page, it decrements the value by one. When the value becomes zero, the printer sets status bit MO%LPC and generates an interrupt if the .MOPSI function was given previously.

If the caller specifies a value of 0 in the register, the system will maintain the page counter and will not generate an interrupt to the caller when the page counter becomes zero.

If the caller specifies a value of -1 in the register, the value will be ignored.

37 .MORST Read the status of the line printer. The status is obtained from the front end, and the caller is blocked until it receives the status.

Argument Block:

Word Contents

0 word count including this word

TOPS-20 MONITOR CALLS
(MTOPR)

1 status word. The following bits are defined:

- B0(MO%LCP) Line printer is a lower case printer. This bit is set only if a .MOSTS function declaring the printer lower case was executed previously.
- B1(MO%RLD) Front end has been reloaded. This bit is reset to zero the next time any I/O activity begins for the line printer.
- B10(MO%FER) A fatal hardware error occurred. This condition generates a software interrupt on channel 11 (.ICDAE).
- B12(MO%EOF) All data sent to printer has actually been printed.
- B13(MO%IOP) Output to the line printer is in progress.
- B14(MO%SER) A software error (for example, interrupt character, page counter overflow) occurred.
- B15(MO%HE) A hardware error occurred. This error generates a software interrupt on channel 11 (.ICDAE). This condition usually requires that the forms be realigned.
- B16(MO%OL) Line printer is offline. This bit is set on the occurrence of any hardware condition that requires operator intervention.
- B17(MO%FNX) Line printer does not exist.
- B30(MO%RPE) A RAM parity error occurred.
- B31(MO%LVU) The line printer has an optical (12-channel tape reader) VFU.
- B33(MO%LVF) A VFU error occurred. The paper has to be realigned.

TOPS-20 MONITOR CALLS
(MTOPR)

B34(MO%LCI) A character interrupt occurred. This generates a software interrupt on channel 11 (.ICDAE).

B35(MO%LPC) The page counter register has overflowed.

Bits 2-17 contain the software status word from the front end, and bits 20-35 contain the hardware status word.

2 value of page counter register. A value of -1 indicates the printer has no page counter value defined.

40 .MOFLO Flush any line printer output that has not yet been printed.

PCDP Functions

The functions available for physical card punches (PCDP) are described below. Like the PLPT functions, these functions accept the address of an argument block in AC3. The first word of the block contains the length (including this word) of the block. Remaining words in the block contain arguments for the particular function.

| Code | Symbol | Meaning |
|------|--------|---|
| 27 | .MOPSI | Enable for a software interrupt on nonfatal device conditions. Examples of these conditions are: <ol style="list-style-type: none"> 1. Device changed from offline to online. 2. Device changed from online to offline. Other device errors or software conditions are not handled by this function; instead they cause a software interrupt on channel 11 (.ICDAE). <p>Argument Block:</p> <p>Word Contents</p> <p>0 word count including this word</p> <p>1 interrupt channel number</p> <p>2 flags. The following flag is defined:</p> <p style="margin-left: 40px;">B0(MO%MSG) Suppress standard CTY device messages.</p> |

TOPS-20 MONITOR CALLS
(MTOPR)

37 .MORST Read the status of the card punch. The status is obtained from the front end, and the caller is blocked until it receives the status.

Argument Block:

Word Contents

| | |
|-------------|--|
| 0 | word count including this word |
| 1 | status word. Bits 2-17 contain the software status word from the front end, and bits 20-35 contain the hardware status word. |
| | |
| B10(MO%FER) | Fatal error condition |
| B12(MO%EOF) | All pending output has been processed |
| B13(MO%IOP) | Output in progress |
| B14(MO%SER) | Software error has occurred (would generate an interrupt on an assigned channel) |
| B15(MO%HE) | Hardware error has occurred (would generate interrupt on channel .ICDAE) |
| B16(MO%OL) | Card punch is offline. This bit is set when operator intervention is required (card jam, hopper empty, or stacker full). |
| B17(MO%FNX) | Card punch doesn't exist |
| B32(MO%HEM) | Hopper is empty or stacker is full |
| B33(MO%SCK) | Stack check |
| B34(MO%PCK) | Pick check |
| B35(MO%RCK) | Read check |

PCDR Functions

The functions available for physical card readers (PCDR) are described below. These functions accept the address of an argument block in AC3. The first word of the block contains the length (including this word) of the block. Remaining words in the block contain arguments for the particular function.

| Code | Symbol | Meaning |
|------|--------|--|
| 27 | .MOPSI | Enable for a software interrupt on nonfatal device conditions. Examples of these conditions are: |
| | | 1. Device changed from offline to online. |

TOPS-20 MONITOR CALLS
(MTOPR)

2. Device changed from online to offline.

Other device errors or software conditions are not handled by this function; instead they cause a software interrupt on channel 11 (.ICDAE).

Argument Block:

Word Contents

- 0 word count including this word
- 1 interrupt channel number
- 2 flags. The following flag is defined:

B0(MO%MSG) Suppress standard CTY device messages.

37 .MORST Read the status of the card reader. The status is obtained from the front end, and the caller is blocked until it receives the status.

Argument Block:

Word Contents

- 0 word count including this word
- 1 status word. B2-17 contain the software status word from the front end, and B20-35 contain the hardware status word.

B0(MO%COL) Card reader is on line. This bit is not obtained from the front end.

B1(MO%RLD) Front end has been reloaded. This bit is reset to zero the next time I/O activity begins for the card reader.

10(MO%FER) A fatal hardware error occurred. This condition generates a software interrupt on channel 11 (.ICDAE).

B12(MO%EOF) Card reader is at end of file.

B13(MO%IOP) Input from the card reader is in progress.

B14(MO%SER) A software error (for example, interrupt character) occurred.

TOPS-20 MONITOR CALLS
(MTOPR)

B15(MO%HE) A fatal hardware error occurred. This error generates a software interrupt on channel 11 (.ICDAE).

B16(MO%OL) Card reader is off line. This bit is set on the occurrence of any hardware condition that requires operator intervention.

B17(MO%FNX) Card reader does not exist.

B31(MO%SFL) The output stacker is full.

B32(MO%HEM) The input hopper is empty.

B33(MO%SCK) A card did not stack correctly in the output stacker.

B34(MO%PCK) The card reader failed to pick a card correctly from the input hopper.

B35(MO%RCK) The card reader detected a read error when reading a card.

PTY Functions

The functions available for pseudo-terminals (PTY) are described below. Some of these functions accept arguments in AC3. (See the individual descriptions.)

| Code | Symbol | Meaning |
|------|--------|---|
| 24 | .MOAPI | Assign PTY interrupt channels. This function requires AC2 to contain: B0(MO%WFI) enable waiting-for-input interrupt B1(MO%OIR) enable output-is-ready interrupt B12-17(MO%SIC) software interrupt channel number for input to the PTY. The channel number used for output from the PTY is one greater than the channel number used for input to the PTY. B18-35 function code |
| 25 | .MOPIH | Determine if PTY job needs input. On a successful return, AC2 contains 0(.MONWI) if PTY job is not waiting for input or contains -1(.MOWFI) if PTY job is waiting for input. |

TOPS-20 MONITOR CALLS
(MTOPR)

26 .MOBAT Set batch control bit. This function requires AC3 to contain 0(.MONCB) if the job is not to be controlled by batch or to contain 1(.MOJCB) if the job is to be controlled by batch. To obtain this value, the process can execute the GETJI JSYS, function .JIBAT.

TTY Functions

| Code | Symbol | Meaning |
|------|--------|---|
| 25 | .MOPIH | Determine if TTY job needs input. On a successful return, AC2 contains 0(.MONWI) if TTY job is not waiting for input or contains -1(.MOWFI) if TTY job is waiting for input. |
| 26 | .MOSPD | Set the terminal line speed. This function accepts in AC3 the desired line speed (input speed in the left half and output speed in the right half). The left half of AC2 contains flag bits indicating the type of line being set. If B0(MO%RMT) is on, the line is a remote (dataset) line. If B1(MO%AUT) is on, the line is a remote autobaud line (is automatically set at 300 baud, and the contents of AC3 are ignored. The process must have WHEEL or OPERATOR capability enabled to set B0(MO%RMT) and B1(MO%AUT). In addition, these bits can only be set at start-up time. They cannot be set during timesharing.) |
| 27 | .MORSP | Return the terminal line speed. On a successful return, left half of AC2 contains flag bits indicating the type of line, and AC3 contains the speed (input speed in the left half and output speed in the right half). If B0(MO%RMT) of AC2 is on, the line is a remote line, and if B1(MO%AUT) is on, the line is a remote autobaud line. AC3 contains the speed or contains -1 if the speed is unknown or is not applicable. |
| 30 | .MORLW | Return the terminal page width. On a successful return, AC3 contains the width. |
| 31 | .MOSLW | Set the terminal page width. This function requires AC3 to contain the desired width. |

TOPS-20 MONITOR CALLS
(MTOPR)

- 32 .MORLL Return the terminal page length. On a successful return, AC3 contains the length.
- 33 .MOSLL Set the terminal page length. This function requires AC3 to contain the desired length.
- 34 .MOSNT Specify if terminal line given in AC1 is to receive system messages. This function requires AC3 to contain 0 (.MOSMY) to allow messages or 1 (.MOSMN) to suppress messages.
- 35 .MORNT Return a code indicating if terminal line given in AC1 is to receive system messages. On a successful return, AC3 contains 0 (.MOSMY) if messages are being sent to this line or 1 (.MOSMN) if messages are being suppressed to this line.
- 36 .MOSIG Specify if input on this terminal line is to be ignored when the line is inactive (is not assigned or opened). This function requires AC3 to contain 0 if characters on this line are not to be ignored or 1 if characters on this line are to be ignored. When input is being ignored and characters are typed, no CTRL/G (bell) is sent, as is the normal case when characters are typed on an inactive line.
- 37 .MORBM Read the 128-character break mask. The argument block (filled in by monitor) is the same as for .MOSBM (below).
- 40 .MOSBM Set the 128-character break mask.
- Argument Block:
- E: 0,,4
E+1-E+4: character mask. The leftmost 32 bits of each consecutive word correspond to the ASCII character set in ascending order. For example, 1B0 in word E+1 (of the argument block) corresponds to ASCII code 000 (null), 1B1 in word E+1 corresponds to ASCII code 001 (SOH). Bits 32-35 of each word must be zero.
- 41 .MORFW Return the current value of the field width in AC3. Note that this may be less than the value last set by .MOSFW. If the field width is set to value X and two characters are read before the .MORFW is executed, the value returned will be X-2. A zero returned in AC3 indicates that no field width is now in effect.

TOPS-20 MONITOR CALLS
(MTOPR)

- 42 .MOSFW Set the field width to the value in AC3. A zero indicates that no field width is in effect.
- 43 .MOXOF Enable/disable pause-at-end-of-page mode. This function controls the TOPS-20 feature that sends exactly n lines of data to the terminal and suspends data transmission (n is the terminal length parameter, set by function .MOSLL). The user may manually resume data transmission by typing ^Q.
- AC3 contains one of the following values:
- 0 .MOOFF Disable pause-at-end-of-page mode
1 .MOONX Enable pause-at-end-of-page mode
- Note that this feature operates independently of the pause-on-command mode implemented in the JFN mode word (see bit TT%PGM of the JFN mode word).
- 44 .MORXO Read the end-of-page mode. This function returns, in AC3, a one if PAUSE ON END-OF-PAGE is set for the terminal, a zero otherwise.
- 45 .MOSLC Set the terminal's line counter to value in AC3. This counter is incremented by the monitor everytime a linefeed is output to the terminal. The monitor clears this counter only when a line becomes active.
- 46 .MORLC Read the terminal's line counter and return with its value in AC3.
- 47 .MOSLM Set line maximum to the value in AC3. This function sets the maximum value of the line counter seen so far. The monitor compares the line counter with the maximum every time a linefeed is typed, and if the line counter value is larger, the monitor sets the line maximum to the value of the line counter. When TEXTI moves the cursor up on screen terminals, it decrements the line counter.
- 50 .MORLM Read the current value of the line maximum and return with its value in AC3.
- 51 .MOTPS Assign terminal interrupt channels. An interrupt will be generated if a character is input, or an output-buffer-empty condition occurs on output.
- AC3 contains the address of a two-word argument block. The first word of the block contains the

TOPS-20 MONITOR CALLS
(MTOPR)

number of words in the block (2), and the second word of the block contains the following: output PSI channel,,input PSI channel. All input or output PSI channels for the terminal are cleared by placing a -1 in the appropriate half, or both halves, of word 2 of the argument block.

- 52 .MOPCS Set the pause and unpause characters for the terminal. This function requires that AC3 contain the pause character in the left half, and the unpause (continue-after-pause) character in the right half. The characters can be the same, but should not be CTRL/Q or CTRL/S.
- 53 .MOPCR Read the terminal pause and unpause (continue-after-pause) characters. This function returns, in AC3, the pause character in the left half, and the unpause character in the right half.
- 54 .MORTF Read the setting of various terminal functions. This function returns the settings in AC3.
B34(MO%NUM) All nonprivileged SENDs are refused.
B35(MO%NTM) All messages are refused.
- 55 .MOSTF Set or clear the setting of various terminal functions. This function accepts the settings in AC3.
B34(MO%NUM) Refuse all nonprivileged SENDs.
B35(MO%NTM) Refuse all messages (SENDs, LINKs, nonprivileged ADVICE, privileged BOUTs and SOUTs). Implements the TERMINAL INHIBIT Command.
- 56 .MOTCE Set two-character escape sequence. This function requires that AC3 contain the 2-character escape sequence, right justified. Neither character can be a null, and the 2 characters cannot be the same.
- 57 .MORTC On return AC3 contains the 2-character escape sequence, right justified.
- 60 .MOCTM This function returns nonzero in AC3 if the terminal is a CTERM terminal:

returns 1 if remote system supports full CTERM functionality

returns 2 or greater if remote system supports limited CTERM functions

TOPS-20 MONITOR CALLS
(MTOPR)

61 .MOTXT Set up for remote TEXTI% call (monitor only).
 Call with AC3 containing flags,,length, where
 flags have the same format as the .RDFLG word in
 the TEXTI% monitor call, and length is the maximum
 length of the read. The following flags are the
 only significant ones:

RD%RIE return if input buffer is empty
RD%RAI raise input
RD%NED disable some editing characters

AC4 contains a byte pointer to ctrl-R buffer; 0 if
no reprompt text.

62 .MOHUP Hangup the terminal line specified. This function
 is used by a program to break the connection on a
 DECnet NRT, DECnet CTERM, TCP/IP TVT, or LAT
 terminal line. On a RSX20F terminal line
 configured as REMOTE, the DTR signal is lowered.

Independent of this MTOPR function, when a program
uses the CLOSF% JSYS to close the last JFN
associated with a terminal line, DTR is lowered.
The terminal line must not be the controlling
terminal for any job and must be an RSX20F
terminal which is configured as REMOTE in
x-CONFIG.CMD. This feature provides an easy way
for a program to control a dial out modem or other
equipment connected to an RSX20F terminal line.

63 .MOUHU Raise DTR on the specified RSX20F terminal line.
 This function is used by a program to raise the
 DTR signal on a terminal line which is connected
 to RSX20F and configured as REMOTE in
 x-CONFIG.CMD.

Independent of this MTOPR function, when a program
uses the OPENF% JSYS to open a JFN on a terminal,
DTR is raised. The terminal line must be an
RSX20F terminal which is configured as REMOTE in
x-CONFIG.CMD, and must not be the controlling
terminal of a job. This feature provides an easy
way for a program to raise DTR on an RSX20F
terminal line to control a dial out modem or other
equipment.

Generates an illegal instruction interrupt on error conditions below.

MTOPR ERROR MNEMONICS:

ANTX01: No more network terminals available

TOPS-20 MONITOR CALLS
(MTOPR)

DCNX8: Invalid network operation
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
DESX9: Invalid operation for this device
DEVX2: Device already assigned to another job
IOX4: End of labels encountered
IOX5: Device or data error
MTOX1: Invalid function
MTOX2: Record size was not set before I/O was done
MTOX3: Function not legal in dump mode
MTOX4: Invalid record size
MTOX5: Invalid hardware data mode for magnetic tape
MTOX6: Invalid magnetic tape density
MTOX7: WHEEL or OPERATOR capability required
MTOX8: Argument block too long
MTOX9: Output still pending
MTOX10: VFU or RAM file cannot be OPENed
MTOX11: Data too large for buffers
MTOX12: Input error or not all data read
MTOX13: Argument block too small
MTOX14: Invalid software interrupt channel number
MTOX15: Device does not have Direct Access (programmable) VFU
MTOX16: VFU or Translation RAM file must be on disk
MTOX17: Device is not on line
MTOX18: Invalid software interrupt channel number
MTOX19: Invalid terminal line width
MTOX20: Invalid terminal line length
MTOX21: Illegal two-character escape sequence
TTYX01: Line is not active

Allows privileged programs to perform various utility functions for magnetic-tape MT: devices. This JSYS differs from the MTOPR JSYS in that the invoking program need not have a JFN on the MT nor need it even have access to the MT. It is used by MOUNTR to declare a volume switch error and by the access-control program (user supplied) to read file and volume labels.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code

TOPS-20 MONITOR CALLS
(MTU%)

AC2: MT unit number

AC3: Address of argument block

RETURNS +1: Always

The functions and associated argument blocks are as follows:

| Code | Symbol | Function |
|------|--------|--|
| 1 | .MTNVV | Declare volume switch error |
| | | Argument Block: |
| | | Word Symbol Contents |
| | | 0 .MTCNT count of words in block |
| | | 1 .MTCOD error code to return to user |
| | | 2 .MTPTR byte pointer to operator response |
| 2 | .MTRAL | Read labels |
| | | Argument Block: |
| | | Word Symbol Contents |
| | | 0 .MTCNT count of words in block |
| | | 1 .MTVL1 byte pointer to area to hold VOL1 label |
| | | 2 .MTVL2 byte pointer to area to hold VOL2 label |
| | | 3 .MTHD1 byte pointer to area to hold HDR1 label |
| | | 4 .MTHD2 byte pointer to area to hold HDR2 label |

If any of the byte pointers is zero, the associated string is not returned.

The label values are always returned without translation. For example, if the tape is an EBCDIC labeled tape, the returned data will be EBCDIC data.

| | | |
|---|--------|--|
| 3 | .MTASI | Return assignment information |
| | | Argument Block: |
| | | Word Symbol Contents |
| | | 0 .MTCNT count of words in block |

TOPS-20 MONITOR CALLS
(MTU%)

1 .MTPHU returned MTA number associated with the MT. If there is no association, .MTNUL is returned.

This function is used by MOUNTR to determine if there are any existing MT to MTA associations.

4 .MTCVV Clear the volume ID for the specified MT unit. This request will fail if the MT is opened or if the volume belongs to a labeled volume set. Requires WHEEL or OPERATOR capability enabled. There is no argument block.

MTU% ERROR MNEMONICS:

ARGX04: Argument block too small
ARGX05: Argument block too long
CAPX1: WHEEL or OPERATOR capability required
DESX1: Invalid source/destination designator
DESX9: Invalid operation for this device
IOX8: Monitor internal error
OPNX1: File is already open
OPNX8: Device is not on line

Performs various IPCF (Inter-Process Communication Facility) functions, such as enabling and disabling PIDs, assigning PIDs, and setting quotas. See the TOPS-20 Monitor Calls User's Guide for an overview and description of the Inter-Process Communication Facility.

RESTRICTIONS: Some functions require WHEEL, OPERATOR, or IPCF capability enabled.

ACCEPTS IN AC1: Length of argument block

AC2: Address of argument block

RETURNS +1: Failure, error code in AC1

+2: Success. Responses from the requested function are returned in the argument block.

The format of the argument block is as follows:

TOPS-20 MONITOR CALLS
(MUTIL)

| Word | Meaning |
|-------------|---|
| 0 | Code of desired function. (See below.) |
| 1 through n | Arguments for the desired function. The arguments, which depend on the function requested, begin in word 1 and are given in the order shown below. Responses from the requested function are returned in these words. |

The available functions, along with their arguments, are described below.

| Code | Symbol | Meaning |
|------|--------|--|
| 1 | .MUENB | <p>Enable the specified PID to receive packets. The PID must have been created by the caller's job. Also, if the calling process was not the creator of the PID, the no-access bit (IP%NOA) must be off in the IPCF packet descriptor block.</p> <p>Argument</p> <p style="padding-left: 40px;">PID</p> |
| 2 | .MUDIS | <p>Disable the specified PID from receiving packets. The PID must have been created by the caller's job. Also, if the calling process was not the creator of the PID, the no-access bit (IP%NOA) must be off in the IPCF packet descriptor block.</p> <p>Argument</p> <p style="padding-left: 40px;">PID</p> |
| 3 | .MUGTI | <p>Return the PID associated with <SYSTEM>INFO. The PID is returned in word 2 of the argument block.</p> <p>Argument</p> <p style="padding-left: 40px;">PID or job number</p> |
| 4 | .MUCPI | <p>Create a private copy of <SYSTEM>INFO for the specified job. The caller must have IPCF capability enabled.</p> <p>Arguments</p> <p style="padding-left: 40px;">PID to be assigned to <SYSTEM>INFO PID or number of job creating private copy</p> |
| 5 | .MUDES | <p>Delete the specified PID. The caller must own the</p> |

TOPS-20 MONITOR CALLS
(MUTIL)

PID being deleted. To obtain ownership of the PID, the caller can first use the .MUCHO function to assign the PID to the caller's job.

Argument

PID

- 6 .MUCRE Creates a PID for the specified process or job. The flags that can be specified are B6(IP%JWP) to make the PID job wide and B7(IP%NOA) to prevent access to PID from other processes. The caller must have IPCF capability enabled if the job number given is not that of the caller. The PID created is returned in word 2 of the argument block. If a job number is specified, the created PID will belong to the top fork of the job.

Argument

flags,,process handle or job number

- 7 .MUSSQ Set send and receive quotas for the specified PID. The caller must have IPCF capability enabled. The new send quota is given in B18-26, and the new receive quota is given in B27-35. The receive quota applies to the specified PID, but the send quota applies to the job to which that PID belongs.

Arguments

PID
new quotas

- 10 .MUCHO Change the job number associated with the specified PID. The caller must have WHEEL capability enabled.

Arguments

PID
new job number or PID belonging to new job

- 11 .MUFOJ Return the job number associated with the specified PID. The job number is returned in word 2 of the argument block.

Argument

PID

TOPS-20 MONITOR CALLS
(MUTIL)

- 12 .MUFJP Return all PIDs associated with the specified job. Two words are returned, starting in word 2 of the argument block, for each PID. The first word is the PID. The second word has B6(IP%JWP) set if the PID is job wide and B7(IP%NOA) set if the PID is not accessible by other processes. The list is terminated by a 0 PID.
- Argument
- job number or PID belonging to that job
- 13 .MUFSQ Return the send and receive quotas for the specified PID. The quotas are returned in word 2 of the argument block with the send quota in B18-26 and the receive quota in B27-35. The receive quota applies to the specified PID, but the send quota applies to the job to which that PID belongs.
- Argument
- PID
- 15 .MUFFP Return all PIDs associated with the same process as that of the specified PID. The list of PIDs returned is in the same format as the list returned for the .MUFJP function (12).
- Argument
- PID
- 16 .MUSPQ Set the maximum number of PIDs allowed for the specified job. The caller must have IPCF capability enabled.
- Arguments
- job number or PID
PID quota
- 17 .MUFPQ Return the maximum number of PIDs allowed for the specified job. The PID quota is returned in word 2 of the argument block.
- Argument
- job number or PID
- 20 .MUQRY Return the Packet Descriptor Block for the next packet in the queue associated with the specified

TOPS-20 MONITOR CALLS
(MUTIL)

PID. An argument of -1 returns the next descriptor block for the process, and an argument of -2 returns the next descriptor block for the job. The descriptor block is returned starting in word 1 of the argument block. The calling process and the process that owns the specified PID must belong to the same job.

Argument

PID

21 .MUAPF Associate the PID with the specified process. The calling process and the process that owns the specified PID must belong to the same job.

Arguments

PID
process handle

22 .MUPIC Place the specified PID on a software interrupt channel. An interrupt is then generated when:

1. The .MUPIC function is issued while the PID has one or more messages in its receive queue.
2. The PID's receive queue changes its state from empty to containing a message. Subsequent entries to a queue that is not empty do not cause an interrupt.

If the channel number is given as -1, the PID is removed from its current channel.

The calling process and the process that owns the specified PID must belong to the same job.

Arguments

PID
channel number

23 .MUDFI Set the PID of <SYSTEM>INFO. An error is given if <SYSTEM>INFO already has a PID. The caller must have IPCF capability enabled.

Argument

PID of <SYSTEM>INFO

TOPS-20 MONITOR CALLS
(MUTIL)

24 .MUSSP Place the specified PID into the system PID table at the given offset. The caller must have WHEEL, OPERATOR, or IPCF capability enabled. See .MURSP for a list of system PIDs.

Arguments

index into system PID table
PID

25 .MURSP Return a PID from the system table. The PID is returned in word 2 of the argument block. The system PID table currently has the following entries:

- 0 .SPIPC Reserved for DEC
- 1 .SPINF PID of <SYSTEM>INFO
- 2 .SPQSR PID of QUASAR
- 3 .SPMDA PID of QSRMDA
- 4 .SPOPR PID of ORION
- 5 .SPNSR PID of NETSER
- 6 .SPCUS PID of CUSTOM APPLICATION (used by QUEUE%)
- 7 .SDIPC PID of DEBUG IPCC (used by QUEUE%)
- 10 .SDINF PID of DEBUG <SYSTEM>INFO (used by QUEUE%)
- 11 .SDQSR PID of DEBUG QUASAR (used by QUEUE%)
- 12 .SDMDA PID of DEBUG QSRMDA (used by QUEUE%)
- 13 .SDOPR PID of DEBUG ORION (used by QUEUE%)
- 14 .SDNSR PID of DEBUG NETSER (used by QUEUE%)
- 15 .SDCUSf PID of DEBUG CUSTOM APPLICATION (used by QUEUE%)

Argument

index into system PID table

26 .MUMPS Return the system-wide maximum packet size. The size is returned in word 1 of the argument block.

27 .MUSKP Set PID to receive deleted PID messages. Allows a controller task to be notified if one of its subordinate tasks crashes. After this function is performed, if the subordinate PID is ever deleted (via RESET or the .MUDES MUTIL function), the monitor will send an IPCF message to the controlling PID notifying it that the subordinate

TOPS-20 MONITOR CALLS
(MUTIL)

PID has been deleted. This message contains
.IPCKP in word 0 and the deleted PID in word 1.

Argument

Source (subordinate) PID
Object (controller) PID

30 .MURKP Return controlling PID for this subordinate PID.

Argument

Source (subordinate) PID
Object (controller) PID (returned)

MUTIL ERROR MNEMONICS:

IPCFX2: No message for this PID
IPCFX3: Data too long for user's buffer
IPCFX4: Receiver's PID invalid
IPCFX5: Receiver's PID disabled
IPCFX6: Send quota exceeded
IPCFX7: Receiver quota exceeded
IPCFX8: IPCF free space exhausted
IPCFX9: Sender's PID invalid
IPCF10: WHEEL capability required
IPCF11: WHEEL or IPCF capability required
IPCF12: No free PID's available
IPCF13: PID quota exceeded
IPCF14: No PID's available to this job
IPCF15: No PID's available to this process
IPCF16: Receive and message data modes do not match
IPCF17: Argument block too small
IPCF18: Invalid MUTIL JSYS function
IPCF19: No PID for [SYSTEM]INFO
IPCF20: Invalid process handle
IPCF21: Invalid job number
IPCF22: Invalid software interrupt channel number
IPCF23: [SYSTEM]INFO already exists
IPCF24: Invalid message size
IPCF25: PID does not belong to this job
IPCF26: PID does not belong to this process
IPCF27: PID is not defined
IPCF28: PID not accessible by this process
IPCF29: PID already being used by another process
IPCF30: job is not logged in
IPCF32: page is not private
IPCF33: invalid index into system PID table
IPCF35: Invalid IPCF quota
IPCF36: PID not assigned on this LCS processor

TOPS-20 MONITOR CALLS
(NI%)

Provides the TOPS-20 user interface to the Ethernet.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Always

NI% JSYS OVERVIEW

The NI% JSYS provides a mechanism for transmitting and receiving data over an Ethernet. A general description of the Ethernet, including the architectural structure, can be found in the

Ethernet Specifications, Version 2.

Portals

Portals are the basic working entity of the NI% JSYS. A portal uniquely identifies a particular user of the Ethernet. In order to transmit and receive data, you must have a portal.

There are two types of portals:

1. Regular (transmit and receive)
2. Information-only

A regular portal includes the following information:

1. PSI channels
2. Ethernet channel number
3. Your protocol type
4. Your enabled multicast addresses
5. List of outstanding transmit and receive buffers
6. Counters

Information-only portals only include PSI channels. They have no protocol type and cannot transmit or receive.

Portal ID

A portal ID is a half-word (18-bit) value that uniquely identifies a portal to the NI% JSYS. Portal IDs are fork-wide unique numbers that

TOPS-20 MONITOR CALLS
(NI%)

start at 1 and increase by 1 for every new portal that is opened. Portal IDs are assigned beginning with the lowest available portal ID.

Protocol Types

The protocol type field, EI%PRO, within word .EIPRO, can have several meanings depending on the value it contains. The possible values are:

| Value | Meaning |
|----------|---|
| 0-177777 | Normal Ethernet protocol types. |
| -1 | Information only. No protocol type is associated with this portal. The portal is able to perform any function except transmit or receive functions. |
| -2 | Promiscuous mode is enabled (receive all Ethernet traffic). No other protocol types can be enabled by any user on the system while promiscuous mode is enabled. |
| -3 | Unknown Protocol Type Queue is assigned to this portal. This queue receives messages that do not match any other enabled protocol types. |

Buffer Descriptor Block

Both receive and transmit buffers are described by one type of block. This block is called a Buffer Descriptor Block. Each block contains all the information pertinent to a single buffer.

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .BXLEN | Length of block (including this word). |
| 1 | .BXNXT | Pointer to next Buffer Descriptor Block. |
| 2 | .BXBSZ | Length of buffer (byte count). (Returns: length of datagram.) |
| 3 | .BXBFA | Byte pointer to beginning of buffer. (Returns: byte pointer to beginning of received data.) |
| 5 | .BXBID | Buffer ID (36-bit value associated with the buffer). |
| 6 | .BXSTA | B0(BX%VAL) This block is valid (return only). B18-35(BX%STA) Status mask (return only). |
| 7 | .BXDAD | Destination Ethernet address. |

TOPS-20 MONITOR CALLS
(NI%)

- 11 .BXSAD Source Ethernet address (return only).
- 13 .BXPPO Protocol type.

A number of receive buffers can be associated with each portal. Receive buffers are queued by the .EIRCV (post a receive buffer) function.

When a datagram is received, the received buffer is put onto an internal monitor receive queue. If this receive queue makes a transition from empty to non-empty, an interrupt is generated on the "receive completion" channel.

The .EIRRQ (read receive queue) function is used for reading the internal receive queue. This function takes a Buffer Descriptor Block chain as an argument. Each block in the chain is filled in with all the information specific to a received buffer. This includes a byte count, a byte pointer, and a buffer ID.

Buffer Descriptor Blocks are chained by placing a pointer in .BXNXT. This capability allows for efficient manipulation of multiple datagram buffers with fewer monitor calls.

Receive Buffer Pointer

The location of a receive buffer is specified by a byte pointer (any format) stored in .BXBFA and .BXBFA+1.

Receive Buffer Size

The size of a receive buffer (in bytes) is specified in .BXBSZ. The size in .BXBSZ depends on whether or not padding is enabled. If padding is not being used with this portal, the buffer size must include room for:

1. User data field from the received datagram (46-1500 (decimal) bytes long).
2. Cyclic Redundancy Check (CRC) (four bytes long).

For example, if the maximum message size for your protocol is 100 bytes, you must use receive buffers that are 104 bytes long.

For portals that use padding, the buffer size must include room for:

1. Data Length Field (two bytes long).
2. User data field from the received datagram (44-1498 (decimal) bytes long).

TOPS-20 MONITOR CALLS
(NI%)

3. Cyclic Redundancy Check (CRC) (four bytes long).

For example, if your protocol specifies that padding should be used, and states that the maximum message size (excluding padding) is 200 bytes long, you must use receive buffers that are 206 bytes long.

NOTE

The minimum receive buffer size is 50 (decimal) bytes,
and the maximum receive buffer size is 1504 (decimal)
bytes.

Received Datagram Pointer

The byte pointer returned in .BXBFA is the same type that was specified when the buffer was originally queued. This byte pointer (any format) points to the first byte of user data. If padding is not in use, the byte pointer is identical to the one that was used to post the buffer. If padding is in use, the byte pointer is advanced past the data length field.

Received Datagram Length

.BXBSZ contains the length of only the data portion of the message (not including the CRC). If padding is in use, .BXBSZ contains the value in the data length field of the padded datagram.

Receive Buffer Constraints

There are a number of constraints on receive buffers:

- o They must be word-aligned.
- o Trailing bytes are indeterminate.

Due to a hardware restriction, the buffer must be word aligned. Therefore, the byte pointer must indicate a word-aligned byte. As an example, byte pointers 441000,,ADDR and 011000,,ADDR-1 are both valid byte pointers to a word-aligned buffer at ADDR.

Note that if the length of the received datagram is not a multiple of four, the trailing bytes, up to the end of the last word, are indeterminate after the buffer is filled. For example, if you specified a length of 41 (decimal) bytes, there is room for three more bytes within the last word of the buffer, and the contents of those bytes are indeterminate.

Transmit Buffers

Transmit buffers are queued to the channel by the .EIXMT (send a

TOPS-20 MONITOR CALLS
(NI%)

datagram) function. Any number of buffers can be queued at a given time. When the channel completes transmission of a buffer, an interrupt is signaled on the "Transmission Complete" interrupt channel. The list of transmitted buffers can be obtained via the .EIRTQ (read transmit queue) function.

.BxBFA and .BxBFA+1 contain a byte pointer (any format) to a buffer, and .BXBSZ contains the length of that buffer (in bytes).

Unlike receive buffers, transmit buffers do not need to be word aligned. The maximum and minimum data lengths depend on whether padding is in use. If padding is in use, the maximum data length is 1498 (decimal) bytes, and the minimum data length is zero. When padding is not in use, the maximum data length is 1500 (decimal) bytes, and the minimum data length is 46 (decimal) bytes.

Channel States

The Ethernet channel participates in a state machine that can be observed and partially controlled by the user.

| Symbol | User Settable | Meaning |
|--------|---------------|---|
| .EISVG | No | Virgin - has never run before |
| .EISRE | Yes | Reload - reload requested |
| .EISCR | No | Cannot reload - reload request timed out |
| .EISIN | No | Init - waiting for response to first command |
| .EISRN | Yes | Run - channel is running and can accept commands |
| .EISDP | Yes | Dump - a dump was requested |
| .EISDR | Yes | Dump and reload - dump followed by a reload request |
| .EISBK | No | Broken - channel cannot be initialized |
| .EISOF | Yes | Off - channel is off |
| .EISRR | Yes | Reload requested - make KNILDR run |

The NI% JSYS also provides a number of other functions for obtaining information and controlling the Ethernet. These are described in the individual function descriptions on the following pages.

All functions use the same general argument block format:

TOPS-20 MONITOR CALLS
(NI%)

| Word | Symbol | Meaning |
|-------------|------------------|---|
| 0 | .EILEN .EIFCN | B0-17 (EI%LEN) Length of argument block B18-35(EI%FCN) Function code (see below) |
| 1 through n | | Arguments for the desired function. The arguments, which depend on the function requested, begin in word 1 and are described as part of the specific function descriptions. |

NOTE

All fields that are not explicitly described in the description for a particular function are ignored by that function.

The following errors are possible on failure from all functions:

CAPX1: WHEEL or OPERATOR capability required
NIEIFC: Illegal Function Code

The available functions are:

| Function Code | Symbol | Meaning |
|---------------|--------|-----------------------------|
| 1 | .EIOPN | Open a portal |
| 2 | .EICLO | Close a portal |
| 3 | .EIRCV | Post a receive buffer |
| 4 | .EIRRQ | Read receive queue |
| 5 | .EIXMT | Transmit datagram(s) |
| 6 | .EIRTQ | Read transmit queue |
| 7 | .EIEMA | Enable a multicast address |
| 10 | .EIDMA | Disable a multicast address |
| 11 | .EIRPL | Read portal list |
| 12 | .EIRCL | Read channel list |
| 13 | .EIRPC | Read portal counters |
| 14 | .EIRCC | Read channel counters |
| 15 | .EIRCI | Read channel information |

TOPS-20 MONITOR CALLS
(NI%)

| | | |
|----|--------|----------------------------------|
| 16 | .EISCS | Set channel state |
| 17 | .EISCA | Set channel address |
| 20 | .EIGET | Obtain ownership of the channel |
| 21 | .EIREL | Release ownership of the channel |
| 22 | .EIRPI | Read portal information |
| 22 | .EIMAX | Maximum function value |

The available functions, along with their arguments, are described below.

Open a Portal - .EIOPN

This function creates portals. It returns a portal ID in .EIPID. The same portal ID must be used in all subsequent calls that are associated with this portal.

The portal is always created, even if the channel is not running (as indicated by .EISTA). This is done so the user can be notified of the channel coming online without having to poll.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--|
| 1 | .EIFLG | B4(EI%PAD) Enable padding feature with this portal |
| | .EIPID | B18-35(EI%PID) Portal ID (return only) |
| 2 | .EICHN | B0-17(EI%CHN) Ethernet channel number |
| | .EIPRO | B18-35(EI%PRO) Protocol type |
| 3 | .EIPSI | B0-11(EI%TCH) Software interrupt channel for notification of transmit complete |
| | | B12-23(EI%RCH) Software interrupt channel for notification of receive complete |
| | | B24-35(EI%SCH) Software interrupt channel for notification of status change |
| 4 | .EISTA | Ethernet channel status (return only) |

The protocol type must not be associated with any other existing portals on the system. It is not possible to transmit or receive on a protocol type that is already assigned.

Fields EI%TCH, EI%RCH, EI%SCH are used to indicate which software

TOPS-20 MONITOR CALLS
(NI%)

interrupt channels should be used to indicate the occurrence of certain events. If an interrupt is not desired for a particular event, -1 should be placed in the field corresponding to that event.

The following errors are possible on failure of this function:

MONX05: Insufficient system resources (no resident free space)
MONX06: Insufficient system resources (no swappable free space)
NIENSC: No such channel
NIEIVP: Illegal value for protocol type field
NIEPIU: Protocol type already in use

Close a Portal - .EICLO

This function closes portals and releases all resources associated with a portal. EI%PID indicates which portal will be closed.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--------------------------|
| 1 | .EIPID | B18-35(EI%PID) Portal ID |

The following error is possible on failure of this function:

NIENSP: No such portal

Post a Receive Buffer - .EIRCV

This function supplies buffers to the channel driver for the asynchronous receipt of datagrams.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B0(EI%BLK) Function should block B1(EI%TBA) Transmit buffer available B2(EI%RBA) Receive buffer available |
| | .EIPID | B18-35(EI%PID) Portal ID |
| 5 | .EIBCP | Address of first Buffer Descriptor Block |

The format of the Buffer Descriptor Block supplied by the user:

| Word | Symbol |
|------|--------|
| 0 | .BXLEN |
| 1 | .BXNXT |

TOPS-20 MONITOR CALLS
(NI%)

2 .BXBSZ
3 .BXBFA
5 .BXBID

The following errors are possible on failure of this function:

NIENSP: No such portal
NIEIFB: Improperly formatted buffer
NIEIBP: Illegal byte pointer
NIEIBS: Illegal buffer size
MONX05: Insufficient system resources (no resident free space)
MONX06: Insufficient system resources (no swappable free space)

Read Receive Queue - .EIRRQ

Each block in the Buffer Descriptor Block chain is filled with data appropriate to a received datagram. This occurs until either there are no more received datagrams, or the chain runs out (that is, .BXNXT contains zero).

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--|
| 1 | .EIFLG | B0(EI%BLK) Function should block until all outstanding receive buffers are filled B1(EI%TBA) Transmit buffer available B2(EI%RBA) Receive buffer available |
| | .EIPID | B18-35(EI%PID) Portal ID |
| 5 | .EIBCP | Address of first Buffer Descriptor Block |

The format of the Buffer Descriptor Block supplied by the user:

| Word | Symbol |
|------|--------|
| 0 | .BXLEN |
| 1 | .BXNXT |

The format of the block returned to the user is:

2 .BXBSZ
3 .BXBFA
5 .BXBID
6 .BXSTA
7 .BXDAD
11 .BXSAD
13 .BXPRO

The buffer ID is the same one supplied in .BXBID when this buffer was posted using the .EIRCV (post a receive buffer) function.

TOPS-20 MONITOR CALLS
(NI%)

The status field, BX%STA, contains zero if the datagram was received successfully; otherwise it contains an error code.

The following errors are possible on failure of this function:

NIERDL: Received datagram too long
NIERAB: Receive aborted
NIELER: Length Error

In the event of a NIERDL: error, .BXBFA points to the portion of the data that fits into the buffer. In this case .BXBSZ contains the "attempted" length, as opposed to the "actual" length of the data. That is, if the datagram was actually 300 bytes, and your buffer was only 200 bytes, then .BXBSZ contains 300. This error cannot occur if padding is enabled.

In the event of a NIELER: error, the data length field is ignored and returned to the user along with the rest of the datagram. .BXBSZ contains the actual length of the datagram (the number of bytes received over the wire).

The protocol type field is only returned when doing promiscuous receives, or when receiving from the "Unknown Protocol Type Queue."

Transmit Datagram(s) - .EIXMT

This function transmits datagrams to the Ethernet address specified in .BXDAD. Each buffer in the Buffer Descriptor Block chain is transmitted in turn, until zero is encountered in .BXNXT.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B1(EI%TBA) Transmit buffer available B2(EI%RBA) Receive buffer available |
| | .EIPID | B18-35(EI%PID) Portal ID |
| 5 | .EIBCP | Address of first Buffer Descriptor Block |

The format of the Buffer Descriptor Block supplied by the user:

| Word | Symbol |
|------|--------|
| 0 | .BXLEN |
| 1 | .BXNXT |
| 2 | .BXBSZ |
| 3 | .BXBFA |
| 5 | .BXBID |
| 7 | .BXDAD |

TOPS-20 MONITOR CALLS
(NI%)

The format of the block returned to the user is:

6 .BXSTA (BX%STA)

The following errors are possible on failure of this function:

MONX05: Insufficient system resources (no resident free space)
MONX06: Insufficient system resources (no swappable free space)
NIENSP: No such portal
NIENPE: No protocol type enabled for this portal
NIEIBS: Illegal buffer size
NIEIBP: Illegal byte pointer

Read Transmit Queue - .EIRTQ

This function returns the data associated with transmitted datagrams. Each transmitted datagram is returned until either there are no more transmitted datagrams, or the Buffer Descriptor Chain runs out (as indicated by 0 in .BXNXT).

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B1(EI%TBA) Transmit buffer available B2(EI%RBA) Receive buffer available |
| | .EIPID | B18-35(EI%PID) Portal ID |
| 5 | .EIBCP | Address of first Buffer Descriptor Block |

The format of the Buffer Descriptor Block supplied by the user:

| Word | Symbol |
|------|--------|
| 0 | .BXLEN |
| 1 | .BXNXT |

The format of the block returned to the user is:

| | |
|---|----------------------------|
| 2 | .BXBSZ |
| 3 | .BXBFA |
| 5 | .BXBID |
| 6 | .BXSTA (BX%VAL and BX%STA) |
| 7 | .BXDAD |

All fields (except BX%STA, the status field) are the same as specified for the .EIXMT (send a datagram) function.

If the transmit was successful the returned status is zero; otherwise an error code appears in field BX%STA of word .BXSTA.

TOPS-20 MONITOR CALLS
(NI%)

The following errors are possible on failure of this function:

NIEDNS: Datagram not sent
NIEEXC: Excessive collisions
NIECCF: Carrier check failed
NIESHT: Short circuit
NIEOPN: Open circuit
NIERFD: Remote failure to defer

Enable a Multicast Address - .EIEMA

This function allows a portal to receive datagrams destined for the Ethernet multicast address specified in .EIBCP. The specified Ethernet address must be a multicast address (the low-order bit of byte 0 of the address must be 1, that is, 1B7).

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--------------------------------------|
| 1 | .EIPID | B18-35(EI%PID) Portal ID |
| 6 | .EIAR1 | Ethernet multicast address (2 words) |

The following errors are possible on failure of this function:

MONX05: Insufficient system resources (no resident free space)
MONX06: Insufficient system resources (no swappable free space)
NIENSP: No such portal
NIENPE: No protocol type enabled for this portal
NIEIMA: Illegal multicast address
NIEIBP: Illegal byte pointer
NIENRE: No room for entry

Disable a Multicast Address - .EIDMA

This function disables a portal from receiving datagrams bound for the multicast address specified in .EIBCP. The specified Ethernet address must be previously enabled using the .EIEMA (enable a multicast address) function.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|----------------------------|
| 1 | .EIPID | B18-35(EI%PID) Portal ID |
| 6 | .EIAR1 | Ethernet multicast address |

The following errors are possible on failure of this function:

TOPS-20 MONITOR CALLS
(NI%)

MONX05: Insufficient system resources (no resident free space)
MONX06: Insufficient system resources (no swappable free space)
NIENSP: No such portal
NIENPE: No protocol type enabled
NIEANE: Address not enabled
NIEIMA: Illegal multicast address
NIEIBP: Illegal byte pointer

Return Portal List - .EIRPL

This function returns a list of all open portals for your fork or for the system.

The list is returned in the buffer pointed to by .EIAR2 in the argument block. Each portal ID occupies a full word, and is right-justified. If the "global" bit (EI%GBL) is set, then the left half of each entry contains the job number that "owns" the portal.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B5(EI%GBL) Return all portal IDs for the system |
| 6 | .EIAR1 | Size of destination buffer |
| 7 | .EIAR2 | Address of destination buffer for portal IDs |

Upon return, the first word of the argument block contains the number of portal IDs returned.

The following error is possible on failure of this function:

NIEIBS: Illegal buffer size

Read Channel List - .EIRCL

This function returns a list of all known Ethernet channels.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--|
| 6 | .EIAR1 | Size of destination buffer |
| 7 | .EIAR2 | Address of destination buffer for channel number |

TOPS-20 MONITOR CALLS
(NI%)

Upon return, the first word of the argument block contains the number of channel IDs returned.

The following error is possible on failure of this function:

NIEIBS: Illegal buffer size

Read Portal Counters - .EIRPC

This function reads (and optionally zeros) portal counters.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B3(EI%ZRO) Zero counters after reading them |
| | | B5(EI%GBL) Use global portal IDs |
| | .EIPID | B18-35(EI%PID) Portal ID |
| 6 | .EIAR1 | Size of block for counters returned |
| 7 | .EIAR2 | Address of block for counters returned |

Counters are only kept for portals that have protocol types associated with them.

The following errors are possible on failure of this function:

MONX05: Insufficient system resources (no resident free space)
MONX06: Insufficient system resources (no swappable free space)
NIENSP: No such portal
NIEIBS: Illegal buffer size

Read Channel Counters - .EIRCC

This function returns (and optionally zeros) the counters associated with a channel.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B3(EI%ZRO) Zero counters after reading them |
| 2 | .EICHN | B0-17(EI%CHN) Channel number |
| 6 | .EIAR1 | Counter buffer size |
| 7 | .EIAR2 | Pointer to counter buffer |

TOPS-20 MONITOR CALLS
(NI%)

The following errors are possible on failure of this function:

NIENSC: No such channel
NIEIBS: Illegal buffer size

Read Channel Information - .EIRCI

This function returns various parameters of the channel.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--|
| 4 | .EISTA | Ethernet channel status B0(EI%RUN) Channel is running B18-26(EI%SST) Channel substate B27-35(EI%EXS) Channel external state |
| 5-6 | .EIPHY | Physical address (current address) |
| 7-10 | .EIHRD | Hardware address |

The address in .EIPHY represents the address to which the channel is currently responding. The address in .EIHRD represents the address that is actually built into the device.

The following error is possible on failure of this function:

NIENSC: No such channel

Set Channel State - .EISCS

This function enables or disables a channel. If the channel is disabled, it is left in a state that can be continued later using the enable mechanism. All functions requiring the channel are queued and executed when the channel is enabled.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|--|
| 2 | .EICHN | B0-17(EI%CHN) Channel number |
| 4 | .EISTA | B18-26(EI%SST) Channel substate; New state |

The following errors are possible on failure of this function:

NIENSC: No such channel
NIECIO: Channel is owned by another fork

TOPS-20 MONITOR CALLS
(NI%)

Set Channel Address - .EISCA

This function sets the physical address associated with a channel.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|------------------------------|
| 2 | .EICHN | B0-17(EI%CHN) Channel number |
| 5-6 | .EIPHY | New channel address |

The address specified in .EIPHY must not be a multicast address.

The following errors are possible on failure of this function:

NIENSC: No such channel
NIEICA: Illegal channel address

Obtain ownership of channel - .EIGET

This function acquires ownership of the KLNI. Only the owner of the KLNI is allowed to alter its state or set its address. If there is no owner, anyone is allowed to execute these functions.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|------------------------------|
| 2 | .EICHN | B0-17(EI%CHN) Channel number |

The following error is possible on failure of this function:

NIECIO: Channel is owned by another fork

Release ownership of channel - .EIREL

This function releases ownership of the KLNI.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|------------------------------|
| 2 | .EICHN | B0-17(EI%CHN) Channel number |

The following errors are possible on failure of this function:

NIECIO: Channel is owned by another fork

TOPS-20 MONITOR CALLS
(NI%)

Read Portal Information - .EIRPI

This function returns all information (except counters) in the portal data base for a given portal.

The format of the argument block is:

| Word | Symbol | Meaning |
|------|--------|---|
| 1 | .EIFLG | B4(EI%PAD) Use padding B5(EI%GBL) Use global portal IDs (supplied by user) |
| | .EIPID | B18-35(EI%PID) Portal ID (supplied by user) |
| 2 | .EIJOB | B0-17(EI%JOB) Only if EI%GBL is set (supplied by user) |
| | .EIPRO | B18-35(EI%PRO) Protocol type |
| 2 | .EICHN | B0-17(EI%CHN) Ethernet channel number (return) |
| 3 | .EIPSI | B0-11(EI%TCH) Software interrupt channel for notification of transmit complete B12-23(EI%RCH) Software interrupt channel for notification of receive complete B24-35(EI%SCH) Software interrupt channel for notification of status change |
| 6 | .EIAR1 | Size of multicast buffer |
| 7 | .EIAR2 | Address of multicast buffer |

EIOXM and EIORC indicate the number of buffers that have not been returned using the Transmit Complete or Receive Complete callbacks.

The multicast address list (only returned if .BXBSZ is nonzero) looks like:

| | | |
|-----------|---|-----------------------------------|
| | | +-----+ |
| | | # returned # set |
| | | +-----+ |
| | | High-order bytes of first address |
| | | +-----+ |
| | | Low-order bytes of first address |
| | | +-----+ |
| n-1 | / | / . / |
| address < | / | / . / |
| pairs \ | / | / . / |
| | | +-----+ |

TOPS-20 MONITOR CALLS
(NI%)

The left half of the first word returned contains the number of multicast addresses actually returned. The right half contains the number of addresses that were set. If the number returned is less than the number set, then the block should be enlarged to hold the number set.

The following error is possible on failure of this function:

NIENSP: No such portal

Read Portal Counters

The counters are returned in a block whose format is:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .EPCNT | Number of words written into this block |
| 1 | .EPSLZ | Seconds since last zeroed |
| 2 | .EPBYR | Bytes received |
| 3 | .EPDGR | Datagrams received |
| 4 | .EPBYS | Bytes sent |
| 5 | .EPDGS | Datagrams sent |
| 6 | .EPUBU | User buffer unavailable |

Read Channel Counters

The counters are returned in a block whose format is:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .ECCNT | Number of words written into this block |
| 1 | .ECSLZ | Seconds since last zeroed |
| 2 | .ECBYR | Bytes received |
| 3 | .ECBYS | Bytes sent |
| 4 | .ECDGR | Datagrams received |
| 5 | .ECDGS | Datagrams sent |

TOPS-20 MONITOR CALLS
(NI%)

| | | |
|----|--------|---|
| 6 | .ECMBR | Multicast bytes received |
| 7 | .ECMDR | Multicast datagrams received |
| 10 | .ECDSM | Datagrams sent, initially deferred |
| 11 | .ECDSL | Datagrams sent, single collision |
| 12 | .ECDSM | Datagrams sent, multiple collisions |
| 13 | .ECSF | Send failures |
| 14 | .ECSFM | Send failure bit mask B24(EC%LOC) Loss of carrier B25(EC%XBP) Transmit buffer parity error B26(EC%RFD) Remote failure to defer B27(EC%XFL) Transmitted frame too long B28(EC%OC) Open circuit B29(EC%SC) Short circuit B30(EC%CCF) Collision detect check failed B31(EC%EXC) Excessive collisions |
| 15 | .ECRF | Receive failure |
| 16 | .ECRFM | Receive failure bit mask B27(EC%FLE) Free list parity error B28(EC%NFB) No free buffers B29(EC%FTL) Frame too long B30(EC%FER) Framing error B31(EC%BCE) Block check error |
| 17 | .ECUFD | Unrecognized frame destination |
| 20 | .ECDOV | Data overrun |
| 21 | .ECSBU | System buffer unavailable |
| 22 | .ECUBU | User buffer unavailable |

Inputs an integer number, with leading spaces ignored. This call terminates on the first character not in the specified radix. If that character is a carriage return followed by a line feed, the line feed is also input.

TOPS-20 MONITOR CALLS
(NIN)

ACCEPTS IN AC1: Source designator

AC3: Radix (2-36) of number being input

RETURNS +1: Failure, error code in AC3, updated string pointer, if pertinent, in AC1

+2: Success, number in AC2 and updated string pointer, if pertinent, in AC1

NIN ERROR MNEMONICS:

IFIXX1: Radix is not in range 2 to 36

IFIXX2: First nonspace character is not a digit

IFIXX3: Overflow (number is equal to or greater than 235)

DESX1: Invalid source/destination designator

DESX2: Terminal is not available to this job

DESX3: JFN is not assigned

DESX5: File is not open

Performs the following network utility functions: set local node name, get local node name, set local node number, get local node number, set loopback port, clear loopback port, and find loopback port.

NOTE

Some of these functions are duplicated in the NTMAN% JSYS, which is preferred. Also, some of the functions can only be used before DECnet initializes.

RESTRICTIONS: Some functions require WHEEL, OPERATOR, or MAINTENANCE capability, or DECnet Phase IV software.

ACCEPTS IN AC1: Function code

AC2: Address of argument block

RETURNS +1: Always. If an error occurs, an illegal instruction trap is generated.

TOPS-20 MONITOR CALLS
(NODE)

The available functions and their argument blocks are described below.

| Code | Symbol | Function | | | | | | |
|------|---------|--|------|--------|----------|---|--------|--|
| 0 | .NDSL N | Set local node name Requires WHEEL or OPERATOR capability. This function can only be used before DECnet initializes. Argument Block: <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">0</td> <td style="vertical-align: top;">.NDNOD</td> <td>Byte pointer to ASCIZ node name.</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .NDNOD | Byte pointer to ASCIZ node name. |
| Word | Symbol | Contents | | | | | | |
| 0 | .NDNOD | Byte pointer to ASCIZ node name. | | | | | | |
| 1 | .NDGLN | Get local node name Argument Block: <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">0</td> <td style="vertical-align: top;">.NDNOD</td> <td>Byte pointer to destination for ASCIZ name of local node.</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .NDNOD | Byte pointer to destination for ASCIZ name of local node. |
| Word | Symbol | Contents | | | | | | |
| 0 | .NDNOD | Byte pointer to destination for ASCIZ name of local node. | | | | | | |
| 2 | .NDSNM | Set local node number Requires WHEEL or OPERATOR capability. This function can only be used before DECnet initializes. Argument Block: <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">0</td> <td style="vertical-align: top;">.NDNOD</td> <td>Number to set (Phase II: $2 < n < 127$; Phase III: from 1 to .NDMAX; Phase IV: from 1 to 1023. Can also include area number (B20-25). If no area number is present the default is 1.)</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .NDNOD | Number to set (Phase II: $2 < n < 127$; Phase III: from 1 to .NDMAX; Phase IV: from 1 to 1023. Can also include area number (B20-25). If no area number is present the default is 1.) |
| Word | Symbol | Contents | | | | | | |
| 0 | .NDNOD | Number to set (Phase II: $2 < n < 127$; Phase III: from 1 to .NDMAX; Phase IV: from 1 to 1023. Can also include area number (B20-25). If no area number is present the default is 1.) | | | | | | |
| 3 | .NDGNM | Get local node number Argument Block: <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">0</td> <td style="vertical-align: top;">.NDNOD</td> <td>Returned node number</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .NDNOD | Returned node number |
| Word | Symbol | Contents | | | | | | |
| 0 | .NDNOD | Returned node number | | | | | | |

TOPS-20 MONITOR CALLS
(NODE)

10 .NDGNT Get network topology.

Reads the system's table of reachable nodes for the local area.

Argument Block:

| Word | Symbol | Contents |
|------|----------|--|
| 0 | .NDNND | Number of words in the argument block in the right half (set by the user on the call) and the number of nodes for which the monitor actually returned data in the left half (set by the monitor on return). |
| 1 | .NDCNT | Number of words in a node block (returned). |
| 2 | .NDBK1 | Addresses of N node blocks (one for each node for which the monitor returned data; returned). |
| | .NDBK1+N | Start of an area into which the monitor sequentially placed node blocks (described below). If there is not enough space to hold all of the information, the NODE JSYS will return as much data as will fit, and then fail with error code ARGX04. (Returned) |

Node Block (Returned):

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .NDNAM | Byte pointer to the ASCIZ node name |
| 1 | .NDSTA | Node state |
| | | Code Symbol Meaning |
| | | 0 .NDSON On |
| | | 1 .NDSOF |
| 2 | .NDNXT | Obsolete (always 0) |
| 3-4 | -- | ASCIZ node name (if node name .LE. 4 characters, Word 4 NOT returned) |

TOPS-20 MONITOR CALLS
(NODE)

- 11 .NDSIC Set topology interrupt channel
- This function is used by a process wishing to be notified that the network topology has changed. The program must do the .NDGNT function to obtain the current topology.
- Topology interrupts can only be given for nodes in the local area. No topology interrupts are given if the system is running as an end node.
- Argument Block:
- | Word | Symbol | Contents |
|------|--------|---|
| 0 | .NDCHN | Channel number on which interrupts are desired. |
- 12 .NDCIC Clear topology interrupt channel
- This function is used to clear the request for interrupt on topology change (set by function .NDSIC).
- 13 .NDGVR Get NSP version number
- Argument Block:
- | Word | Symbol | Contents |
|------|--------|---|
| 0 | .NDNVR | Number of versions returned |
| 1 | .NDCVR | Address of a block in which the NSP communications version will be returned. (Block format is shown below.) |
| 2 | .NDRVR | Address of a block in which the NSP routing version will be returned. (Block format is shown below.) |
- Version Block:
- | Word | Symbol | Contents |
|------|--------|-----------------------|
| 0 | .NDVER | Version number |
| 1 | .NDECO | ECO number |
| 2 | .NDCST | Customer change order |

TOPS-20 MONITOR CALLS
(NODE)

14 .NDGLI Obsolete. See the NTMAN JSYS description for information on lines known to NSP.

15 .NDVFY Verify node name

This function indicates whether the node name supplied by the user is in the monitor's database of known nodes, and if that node can be reached currently.

Argument Block:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .NDNOD | Byte pointer to ASCIZ node name to be checked. |
| 1 | .NDFLG | Flags returned by monitor. |

Flags:

- B0(ND%EXM) The specified node exactly matches a node name in the monitor's node database.
- B1(ND%LGL) The node name is a legal node name.
- B2(ND%RCH) This node is reachable.
- B3(ND%RUK) The reachability of this node is unknown because it is not in this system's network area, or the local node is an end node (non-routing).

16 .NDRNM Return a node name.

This function converts a node number to a node name. (TOPS-20, Version 5.1 only)

Argument Block:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .NDNOD | The node number |
| 1 | .NDCVR | Byte pointer to area where the ASCIZ node name is to be returned. |

17 .NDCIN Return connection information.

TOPS-20 MONITOR CALLS
(NODE)

NOTE

This function is primarily intended for system use. The information returned may change in a future release.

This function returns information about a connection. To use this function, call the first time with words NB.JOB and NB.CHN containing zero. The call returns information about the first connection of the first job with a connection on the system. Subsequent calls report the status of other channels in the job, or, if all channels have been reported, will advance the job number (NB.JOB) until information about all jobs and channels has been returned. A NODX11 error (job number out of range) is returned and NB.JOB is set to -1 after all jobs and channels have been examined.

Special jobs that have connections (NRT or CTERM) are identified by having NB.JOB set to the ASCII name of the channel.

The number of words requested must be at least NB.LEN.

Argument Block:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | NB.RTW | B0-17 (NBRTW) number of words returned |
| 0 | NB.RQW | B18-35 (NBRQW) number of words requested |
| 1 | NB.JOB | Job number, or -1 for no more jobs |
| 2 | NB.CHN | Channel number of connection |
| 3 | NB.OBJ | B0-17 (NBOBJ) receiver object type, or -1 |
| 3 | NB.STA | B18-23 (NBSTA) session control (link) state |
| 3 | NB.XFL | B24-26 (NBXFL) transmit flow control option |
| 3 | NB.RFL | B27-29 (NBRFL) receive flow control option |
| 4 | NB.GOL | B0-17 (NBGOL) receive data request goal |
| 4 | NB.INQ | B18-35 (NBINQ) input quota for link |

TOPS-20 MONITOR CALLS
(NODE)

| | | |
|----|--------|---|
| 5 | NB.OTQ | B0-17 (NBOTQ) output quota for link |
| 5 | NB.DNA | B18-35 (NBDNA) destination node address (remote host name) |
| 6 | NB.SSZ | B0-17 (NBSSZ) segment size (byte count in segment) |
| 6 | NB.RSN | B18-35 (NBRSN) reason for disconnect or reject |
| 7 | NB.LLA | B0-17 (NBLLA) local link address |
| 7 | NB.RLA | B18-35 (NBRLA) remote link address |
| 10 | NB.PKS | B0-17 (NBPKS) packets sent |
| 10 | NB.PKR | B18-35 (NBPKR) packets received |
| 11 | NB.TYP | B0 (NB TYP) 0 means passive connection; 1 means active connection |
| 11 | NB.VER | B1-3 (NBVER) version of remote NSP (0=3.2, 1=3.1, 2=4.0) |
| 11 | NB.JFN | B4-16 (NB JFN) JFN associated with channel |
| 11 | NB.FRK | B18-35 (NBFRK) process number for channel |

20 .NDRDB Read DECnet data blocks

NOTE

This function is primarily intended for system use. The information returned may change in a future release.

Argument Block:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .NDRBT | Type of table to return 1(.NDBSJ) session job 2(.NDBSL) session line 3(.NDBEL) end-user layer link 5(.NDBCT) CTERM data block |
| 1 | .NDRBD | Destination of data |
| 2 | .NDRBJ | First argument for locating table |
| 3 | .NDRBC | Second argument for locating table |

21 .NDSDP Set DECnet initialization parameters

Argument Block:

TOPS-20 MONITOR CALLS
(NODE)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .NDPRM | type of parameter to set 0(.NDRTR) routing type 1(.NDMXA) maximum address 2(.NDMXB) maximum buffers 3(.NDDBL) default buffers per link 4(.NDBSZ) buffer size 5(.NDFLO) flow control |
| 1 | .NDVAL | Value of parameter. This value is dependent on the functions being performed. The following are valid function values: 0(FCM.NO) no flow control (only if .NDFLO is specified) 1(FCM.SG) segment flow control (only if .NDFLO is specified) 4(RNT.L1) level-1 router (only if .NDRTR is specified) 5(RNT.NR) non-routing (only if .NDRTR is specified) |

22 .NDINT Insert node table

Argument Block:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .NDNNN | Number of node definitions |
| 1 | .NDNTA | Address of node table consisting of the number of word pairs specified by .NDNNN. Each word pair is in the following format: word 0 node name in SIXBIT word 1 16 bit node address |

NODE ERROR MNEMONICS:

ARGX02: Invalid function
 ARGX04: Argument block too small
 ARGX19: Invalid unit number
 CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required
 COMX19: Too many characters in node name
 COMX20: Invalid node name
 MONX06: Insufficient system resources (No swappable free space)
 NODX02: Line not turned off

TOPS-20 MONITOR CALLS
(NODE)

NODX03: Another line already looped
NODX04: No local node name defined
NODX05: Function no longer supported
NODX06: Resource allocation failure
NODX07: Argument block not long enough
NODX10: Channel number out of range
NODX11: Job number out of range
NODX12: Bad table designator
NODX13: Bad 1st argument
NODX14: Bad 2nd argument
NODX15: No such table
NODX16: DECnet is already initialized
NODX17: Illegal parameter value
NSPX25: Illegal DECnet node number
NSPX26: Table of topology watchers is full

Outputs an integer number.

ACCEPTS IN AC1: Destination designator

AC2: Number to be output

AC3: B0(NO%MAG) Output the magnitude. That is, output the number as an unsigned 36-bit number (for example, output -1 as 777777 777777).

B1(NO%SGN) Output a plus sign for a positive number.

B2(NO%LFL) Output leading filler. If this bit is not set, trailing filler is output, and bit 3(NO%ZRO) is ignored.

B3(NO%ZRO) Output 0's as the leading filler if the specified number of columns (NO%COL) allows filling. If this bit is not set, blanks are output as leading filler if the number of columns allows filling.

B4(NO%OOV) Output on column overflow and return an error. If this bit is not set, column overflow is not output.

B5(NO%AST) Output asterisks on column overflow. If

TOPS-20 MONITOR CALLS
(NOUT)

 this bit is not set and bit 4 (NO%OOV) is set, all necessary digits are output on column overflow.

B11-17 Number of columns (including sign column) to output. If this field is 0, as many columns as necessary are output.
(NO%COL)

B18-35 Radix (2-36) of number being output
(NO%RDY)

RETURNS +1: Failure, error code in AC3
 +2: Success, updated string pointer in AC1, if pertinent

NOUT ERROR MNEMONICS:

NOUTX1: Radix is not in range 2 to 36
NOUTX2: Column overflow
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Returns generic network information.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Always

The following function is available:

| Function Symbol | Meaning |
|-------------------|---|
| 0 .NWRRH | Returns information about the originating host of a job. Can also be used to return the terminal line type for network and non-network terminals. |

TOPS-20 MONITOR CALLS
(NTINF%)

NOTE

If an incoming DECnet connection is routed through a node that explicitly specifies routing information (poor man's router), the name of that router node is given, not the name of the node where the terminal is located.

Correct set up of the argument block requires the argument block count, function code, device designator, and the byte pointer. All other fields are filled in upon return.

The argument block must be at least 7 words in length (.NWNUI+2).

The format of the argument block is:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .NWABC | Count of words in argument block (including this word). |
| 1 | .NWFNC | Function code |
| 2 | .NWLIN | TTY device designator; job number or -1 for this job. |
| 3 | .NWNNP | Destination designator; byte pointer to location for monitor to write the name and username, if possible, of the originating node in user address space. For CTERM terminals, the monitor will return NODE::USER. |
| 4 | .NWTTF | Terminal type and flags (Returned) |
| | | B0-8 Flags |
| | | B0(NW%NNN) No node name known |
| | | B9-17 Network type |
| | | 0 NW%NNT non-network terminal |
| | | 1 NW%TCP Internet TCP |
| | | 2 NW%DNA DECnet |
| | | 3 NW%LAT Local Area Terminal (LAT) |

TOPS-20 MONITOR CALLS
(NTINF%)

B18-35 Line type

| | | |
|---|--------|----------------------------|
| 0 | NW%UND | undefined terminal type |
| 1 | NW%FE | front end terminal |
| 2 | NW%PT | pseudo terminal |
| 3 | NW%MC | NRT terminal |
| 4 | NW%TV | TVT terminal |
| 5 | NW%CH | CTERM terminal |
| 6 | NW%LH | LAT terminal |

| | | |
|---|--------|--|
| 5 | .NWNNU | Node number word 1 (Returned) |
| 6 | .NWNUL | Node number word 2 (word 2 is only used for Ethernet addresses with LAT terminals). (Returned) |

NTINF% ERROR MNEMONICS:

ARGX02: Invalid function
ARGX04: Argument block too small
GTJIX2: Invalid terminal line number
GTJIX3: Invalid job number
GTJIX4: No such job
TTYX01: Line is not active
TTYX04: Job is detached

NOTE

This JSYS is primarily intended for system use. The information returned may change in a future release.

Provides an interface between the DECnet-20 Network Management layer and lower layers of the DIGITAL Network Architecture.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Always

TOPS-20 MONITOR CALLS
(NTMAN%)

NOTE

Users of the NTMAN% JSYS should be familiar with the Network Management Specification.

Format of Argument Block:

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .NTCNT | Number of words in this argument block |
| 1 | .NTENT | Entity on which to perform function |
| | | Code Symbol Meaning |
| | | 0 .NTNOD Node |
| | | 1 .NTLIN Line |
| | | 2 .NTLOG Logging |
| | | 3 .NTCKT Circuit |
| | | 4 .NTMOD Module |
| | | 5 .NTARE Area |
| 2 | .NTEID | Byte pointer to Entity ID. (See the Network Management Specification for format.) |
| 3 | .NTFNC | Function to be performed |
| | | Code Symbol Meaning |
| | | -4 .NTSLM Set global logging mask |
| | | -3 .NTPSI Set PSI channel for reading events |
| | | -2 .NTMAP Map node number/node name |
| | | -1 .NTREX Return the local node ID |
| | | 0 .NTSET Set Parameter |
| | | 1 .NTCLR Clear Parameter |
| | | 2 .NTZRO Zero all Counters |
| | | 3 .NTSHO Show selected Items |
| | | 4 .NTSZC Show and Zero All Counters |
| | | 5 .NTRET Return List of Items |
| | | 6 .NTEVQ Process the event queue |
| 4 | .NTSEL | Selection criterion for function |
| | | Selectors for Show Selected Items (.NTSHO) |
| | | Code Symbol Meaning |
| | | 0 .NTSUM Summary |
| | | 1 .NTSTA Status |

TOPS-20 MONITOR CALLS
(NTMAN%)

| | | |
|---|--------|-----------------|
| 2 | .NTCHA | Characteristics |
| 3 | .NTCOU | Counters |
| 4 | .NTEVT | Event |
| 5 | .NTCST | Circuit state |

Selectors for Return List of Items (.NTRET)

| Code | Symbol | Meaning |
|------|--------|--|
| -1 | .NTKNO | Known Items |
| -2 | .NTACT | Active Items |
| -3 | .NTLOP | Loop |
| -4 | .NTADJ | Adjacent items |
| -5 | .NTSGN | Significant items |
| 5 | .NTQUA | Byte pointer to function to qualifier |
| 6 | .NTBPT | Byte pointer to parameter data buffer. Pointer is updated to next available byte on return. |
| 7 | .NTBYT | Parameter data buffer length in bytes. Written in buffer for functions .NTMAP, .NTRET, .NTREX, .NTSHO, and .NTSZC. |
| 10 | .NTERR | Network Management return code. (See the Network Management Specification for codes.) |

NTMAN% ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required
 ARGX09: Invalid byte size
 ARGX17: Invalid argument block length
 NTMX1: Network Management unable to complete request

Converts the internal date and time format into separate numbers for local weekday, day, month, year, and time and does not convert the numbers to text. (See Section 2.9.2 for more information.) The ODCNV call gives the caller option of explicitly specifying the time zone and daylight savings time.

ACCEPTS IN AC2: Internal date and time, or -1 for current date and time

TOPS-20 MONITOR CALLS
(ODCNV)

AC4: B0(IC%DSA) Apply daylight savings according to the setting of B1(IC%ADS). If B0 is off, daylight savings is applied only if appropriate for date.

B1(IC%ADS) Apply daylight savings if B0(IC%DSA) is on.

B2(IC%UTZ) Use time zone in B12-17(IC%TMZ). If this bit is off, the local time zone is used.

B3(IC%JUD) Apply Julian day format (Jan 1 is day 1 in conversion)

B12-17 Time zone to use if B2(IC%UTZ) is on.
(IC%TMZ)

B18-35 Local time in seconds since midnight.
(IC%TIM)

RETURNS +1: Always, with

AC2 containing the year in the left half, and the numerical month (0= January) in the right half.

AC3 containing the day of the month (0= first day) in the left half, and the day of the week (0= Monday) in the right half.

AC4 containing

B0 and B2 On for compatibility with the IDCNV call
B1(IC%ADS) On if daylight savings was applied
B3(IC%JUD) On if Julian day format was applied
B12-17 Time zone used
(IC%TMZ)
B18-35 Local time in seconds since midnight
(IC%TIM)

If IC%JUD is set, the Julian day (1=Jan 1, 365=non-leap Dec 31, 366=leap Dec 31, etc) is returned in the right half of AC2 and the left half of AC3 is set to zero.

ODCNV ERROR MNEMONICS:

DATEX6: System date and time are not set
TIMEX1: Time cannot be greater than 24 hours
ZONEX1: Time zone out of range

TOPS-20 MONITOR CALLS
(ODTIM)

Outputs the date and time by converting the internal format of the date and/or time to text. (See Section 2.9.2.)

ACCEPTS IN AC1: Destination designator

AC2: Internal date and time, or -1 for current date and time

AC3: Format option flags (see below), 0 is the normal case

RETURNS +1: Always, with updated string pointer in AC1, if pertinent

The format option flags in AC3 indicate the format in which the date and time are to be output.

ODTIM Option Flags

B0(OT%NDA) Do not output the date and ignore B1-8.

B1(OT%DAY) Output the day of the week according to the format specified by B2(OT%FDY).

B2(OT%FDY) Output the full text for the day of the week. If this bit is off, the 3-letter abbreviation of the day of the week is output.

B3(OT%NMN) Output the month as numeric and ignore B4(OT%FMN).

B4(OT%FMN) Output the full text for the month. If this bit is off, the 3-letter abbreviation of the month is output.

B5(OT%4YR) Output the year as a 4-digit number. If this bit is off, the year is output as a 2-digit number if between 1900 and 1999.

B6(OT%DAM) Output the day of the month after the month. If this bit is off, the day is output before the month.

B7(OT%SPA) Output the date with spaces between the items (for example, 6 Feb 76). If B6(OT%DAM) is also on, a comma is output after the day of the month (for example, Feb 6, 76).

B8(OT%SLA) Output the date with slashes (for example, 2/6/76).

If B7-8 are both off, the date is output with dashes between the items (for example, 6-Feb-76).

TOPS-20 MONITOR CALLS
(ODTIM)

- B9(OT%NTM) Do not output the time and ignore B10-13.
- B10(OT%NSC) Do not output the seconds. If this bit is off, the seconds are output, preceded by a colon.
- B11(OT%12H) Output the time in 12-hour format with AM or PM following the time. If this bit is off, the time is output in 24-hour format.
- B12(OT%NCO) Output the time without a colon between the hours and minutes.
- B13(OT%TMZ) Output the time and follow it with a "-" and a time zone (for example, -EDT).
- B17(OT%SCL) Suppress columnation of the date and time by omitting leading spaces and zeros. This produces appropriate output for a message. If this bit is off, the date and time are output in columns of constant width regardless of the particular date or time. However, full texts of months and weekdays are not columnated. This output is appropriate for tables.
- B35(OT%822) Output time in RFC822 format.

If AC3 is 0, the ODTIM call outputs the date and time in columns in the format

dd-mmm-yy hh:mm:ss

For example, 6-Feb-76 15:14:03.

If AC3 is -1, the ODTIM call interprets the contents as if B1-2, B4-7, and B17 were on (AC3=336001000000) and outputs the date and time in the format

weekday, month day, year hh:mm:ss

as in Friday, February 6, 1976 15:14:03

Additional examples are:

| Contents of AC3 | Typical Text |
|-----------------|-------------------------------------|
| 202201000000 | Fri 6 Feb 76 1:06 |
| 336321000000 | Friday, February 6, 1976 1:06AM-EST |
| 041041000000 | 6/2/76 106:03 |
| 041040000000 | 6/02/76 106:03 |

TOPS-20 MONITOR CALLS
(ODTIM)

ODTIM ERROR MNEMONICS:

DATEX6: System date and time are not set
TIMEX1: Time cannot be greater than 24 hours

All I/O errors are also possible. These errors cause software interrupts or process terminations as described for the BOUT call description.

Outputs the date and/or the time as separate numbers for local year, month, day, or time. (See Section 2.9.2.) This JSYS is a subset of the ODTIM call because the output of dates and times not stored in internal format is permitted. Also, the caller has control over the time and zone printed.

ACCEPTS IN AC1: Destination designator

AC2: Year in the left half, and numerical month (0= January) in the right half

AC3: Day of the month (0= first day) in the left half, and day of the week (0= Monday), if desired, in the right half

AC4: B1(IC%ADS) Apply daylight savings on output
B12-17(IC%TMZ) Time zone in which to output
B18-35(IC%TIM) Local time in seconds since midnight

AC5: Format option flags (see ODTIM for the description of these flags)

NOTE

The only time zones that can be output by B13(OT%TMZ) are Greenwich and USA zones.

RETURNS +1: Always, with updated string pointer in AC1, if pertinent.

TOPS-20 MONITOR CALLS
(ODTNC)

ODTNC ERROR MNEMONICS:

DATEX1: Year out of range
DATEX2: Month is not less than 12
DATEX3: Day of month too large
DATEX4: Day of week is not less than 7
ZONEX1: Time zone out or range
ODTNX1: Time zone must be USA or Greenwich

All I/O errors can occur. These errors cause software interrupts or process terminations as described for the BOUT call description.

Opens the given file. See the TOPS-20 Monitor Calls User's Guide for the explanations of the types of access allowed to a file.

ACCEPTS IN AC1: JFN (right half of AC1) of the file being opened.

AC2: B0-5(OF%BSZ) Byte size (maximum of 36 decimal). If a zero byte size is supplied, the byte size defaults to 36 bits.

B6-9(OF%MOD) Data mode in which to open file.
Common data modes are:

| Code | Symbol | Mode |
|------|--------|----------------|
| 0 | .GSNRM | Normal (ASCII) |
| 1 | .GSSMB | Small buffer |
| 10 | .GSIMG | Image |
| 17 | .GSDMP | Dump |

TCP/IP data modes:

| Code | Symbol | Meaning |
|------|--------|---------------------------|
| 1 | .TCMWI | Interactive |
| 2 | .TCMWH | High throughput |
| 3 | .TCMII | Immediate return |
| 4 | .TCMIH | Buffered immediate return |

(See Section 2.5 for more information on software data modes.)

TOPS-20 MONITOR CALLS
(OPENF)

Useful modes for common devices are:

| Device | Data Modes |
|-------------|---|
| Disk | .GSNRM |
| Card Reader | .GSNRM, .GSIMG |
| Card Punch | .GSNRM, .GSIMG |
| PTY | .GSNRM (PTY receives data in mode of its TTY) |
| Mag Tape | .GSNRM, .GSDMP |
| TTY | .GSNRM, .GSIMG |

B18(OFF%HER) Halt on I/O device or data error. If this bit is on and a condition occurs that causes an I/O device or data error interrupt, the process will instead be halted, and an illegal instruction interrupt will be generated. If bit is off and the condition occurs, the interrupt is generated on its normally-assigned channel. This bit remains in affect for the entire time that the file is open.

B19(OFF%RD) Allow read access.

B20(OFF%WR) Allow write access.

B21(OFF%EX) Allow execute access.

B22(OFF%APP) Allow append access.

B23(OFF%RDU) Allow unrestricted read access. This bit allows you to open a file for reading regardless of simultaneous thawed or frozen openings of the file for reading or writing by other processes or the process executing this call. You can use this bit only if you do not use the OFF%THW or OFF%WR bits.

B25(OFF%THW) Allow thawed access. If this bit is off, the file is opened for frozen access.

Frozen access means there can be only one writer of the file; thawed access means there can be many writers of the file. A program manipulating a thawed file must take into account the fact that other programs may open and modify

TOPS-20 MONITOR CALLS
(OPENF)

that file. Thawed/frozen access has no direct effect on readers of the file, but it does have the indirect effect that is described in the next paragraph.

The first open of a file sets the precedent for future opens: if the first open is thawed, then all subsequent opens must be thawed, regardless if read or write access is desired. The same holds true for frozen access. This condition is in effect until the last close of the file.

See the descriptions of bits OF%DUD and OF%RDU for the interaction of OF%THW with those bits. Also, see the description of the PMAP JSYS for the interaction of PMAP bit PM%ABT with OF%DUD.

B26(OF%AWT) Block program and print a message on the job's terminal if access to file cannot be permitted. The program is blocked until access is granted.

B27(OF%PDT) Do not update access dates of the file.

B28(OF%NWT) Return an error if access to file cannot be permitted.

If B26 and B28 are both off, the default is to return an error if access to the file cannot be granted.

B29(OF%RTD) Enforce restricted access. No other JFN in the system can be opened with this file until the current JFN is released. This bit requires that the user have the ability to set WRITE access to the file.

B30(OF%PLN) Disable line number checking and consider a line number as 5 characters of text.

B31(OF%DUD) Suppress the system updating of modified pages in memory to thawed files on disk. This bit is ignored for

TOPS-20 MONITOR CALLS
(OPENF)

new files, and for files on structures that are shared under CFS-20.

Ordinarily, TOPS-20 updates modified memory pages to disk approximately once each minute. OF%DUD prohibits this automatic update. However, there are two sources of "manual" updating that are not controlled by OF%DUD:

1. A CLOSF JSYS is performed
2. A UFPGS JSYS is performed

OF%DUD and OF%THW interact in the following ways:

| OF%THW | OF%DUD | Effect |
|--------|--------|-------------------------------------|
| 0 | 0/1 | OF%DUD ignored |
| 1 | 0 | Perform automatic file page update |
| 1 | 1 | Suppress automatic file page update |

B32(OF%OFL) Open the device even if it is off-line.

B33(OF%FDT) Force an update of the .FBREF date and time (last read) in the FDB. Also, increment right halfword (number of file references) of .FBCNT count word in the FDB.

B34(OF%RAR) Wait if the file is offline.

RETURNS +1: Failure, error code in AC1

+2: Success

Even though each type of desired file access can be indicated by a separate bit, some accesses are implied when specific bits are set. For example, the setting of the write access bit implies read access if the process is allowed to read the file according to the file's access code. However, if an existing file is opened and only write access is specified (only OF%WR is set), contents of the file are deleted, and the file is considered empty. Thus, to update an existing file, both OF%RD and OF%WR must be set.

Note that if OF%RD, OF%WR, and OF%APP are all zero, OPENF will generate an error. OPENF works as follows for archived and migrated files:

TOPS-20 MONITOR CALLS
(OPENF)

Archived

| | | |
|--------------|--------|-----------|
| OPENF Access | Online | Offline |
| Read | Ok | Fail/Wait |
| Write | Fail | Fail |
| Append | Fail | Fail |

Migrated

| | | |
|--------------|-------------------------|--------------------------------|
| OPENF Access | Online | Offline |
| Read | Ok | Fail/Wait |
| Write | Ok (discard implied) | |
| Append | Ok (discard implied) | Fail/Wait (discard implied) |

The failure cases return an error message (OPNXnn). The fail/wait cases return an error for failure or wait until the OPENF can be successfully completed.

The settings of OF%NWT (never wait for file restore) and OF%RAR (retrieve file if necessary) determine whether a failure or wait occurs. If OF%NWT is set on the OPENF call, OPENF always fails (in the fail/wait cases). If OF%RAR or the job default (See the SETJB monitor call.) is set, the OPENF will wait for the file to be retrieved, and then complete successfully. In the Ok (discard implied) cases, tape pointers for the file, if any, are discarded.

The CLOSF monitor call can be used to close a specific file.

OPENF ERROR MNEMONICS:

- OPNX1: File is already open
- OPNX2: File does not exist
- OPNX3: Read access required
- OPNX4: Write access required
- OPNX5: Execute access required
- OPNX6: Append access required
- OPNX7: Device already assigned to another job
- OPNX8: Device is not on line
- OPNX9: Invalid simultaneous access
- OPNX10: Entire file structure full
- OPNX12: List access required
- OPNX13: Invalid access requested
- OPNX14: Invalid mode requested
- OPNX15: Read/write access required
- OPNX16: File has bad index block

TOPS-20 MONITOR CALLS
(OPENF)

OPNX17: No room in job for long file page table
OPNX18: Unit Record Devices are not available
OPNX23: Disk quota exceeded
OPNX25: Device is write-locked
OPNX26: Illegal to open a string pointer
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
SFBSX2: Invalid byte size
STRX10: Structure is offline
TTYX01: Line is not active
TCPXX1: No IP free space for TCB
TCPX17: Illegal IO mode for TCP device
TCPX18: Illegal byte size for TCP device
TCPX19: TCP connection allready exists
TCPX20: Maximum TCP connections exceeded
TCPX25: Open failure
TCPX30: Illegal TCP IO mode
TCPX31: Connection error or connection rejected
TCPX32: Retransmission timeout
TCPX33: Connection closed or closing

Inputs the next sequential byte from the primary input designator. This call is equivalent to a BIN call with the source designator given as .PRIIN.

RETURNS +1: Always, with the byte right-justified in AC1

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

PBIN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX5: File is not open
IOX1: File is not open for reading
IOX4: End of file reached
IOX5: Device or data error

TOPS-20 MONITOR CALLS
(PABOUT)

Outputs a byte sequentially to the primary output designator. This call is equivalent to a BOUT call with the destination designator given as .PRIOU.

ACCEPTS IN AC1: Byte to be output, right-justified

RETURNS +1: Always

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

PABOUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX2: File is not open for writing
IOX5: Device or data error
IOX6: Illegal to write beyond absolute end of file
IOX11: Quota exceeded
IOX34: Disk full
IOX35: unable to allocate disk - structure damaged

Manipulates program data vectors (PDVs), which begin at program data vector addresses (PDVAs). Program data vectors are used to allow user programs to obtain information about execute-only programs.

ACCEPTS IN AC1: Function code

AC2: Address of the argument block

AC3: Byte pointer to a string in memory

RETURNS +1: Always, with data returned in the data block, an updated count in .POCT2 if needed.

The following describes the format of the argument block to which the address in AC2 points.

TOPS-20 MONITOR CALLS
(PDVOF%)

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .POCT1 | Count 1, the number of words in the argument block. |
| 1 | .POPHD | Handle of the process that the call is to affect |
| 2 | .POCT2 | Count 2, the number of words in the data block. The call returns two counts in this word. The left half contains the number of words of data available for the call to return, and the right half contains the number of words the call did return in the data block. If the right half is smaller than the left half, the call could not return all the data available due to a lack of room in the data block. |
| 3 | .PODAT | Starting address of the data block into which the call returns data |
| 4 | .POADR | Starting address of the range of memory |
| 5 | .POADE | Ending address of the range of memory |

The format of a program data vector is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .PVCNT | Length of the PDV (including this word). |
| 1 | .PVNAM | The address of the name of the program for which this data vector exists. The name is in ASCII representation. (In most cases, a byte pointer should be created to access this string.) |
| 2 | .PVEXP | Address of the exported information vector. |
| 3 | .PVREE | Reserved for DIGITAL. |
| 4 | .PVVER | Program version number. |
| 5 | .PVMEM | Address of a block of memory that contains data describing the program's address space (a memory map). See the LINK manual, Appendix C, for a description of this block. |
| 6 | .PVSYM | Address of the program symbol vector. |
| 7 | .PVCTM | Time at which the program was compiled. |
| 10 | .PVCVR | Version number of the compiler. |
| 11 | .PVLTM | Time at which the program was loaded. |
| 12 | .PVLVR | Version number of LINK. |
| 13 | .PVMON | Address of a monitor data block. (Not currently used.) |
| 14 | .PVPRG | Address of a program data block. (Not currently used.) |
| 15 | .PVCST | Address of a customer-defined data block. |

TOPS-20 MONITOR CALLS
(PDVOF%)

Functions that require a range of memory locations (.POGET and .POREM) interpret words .POADR and .POADE as follows:

- o If .POADR and .POADE are both nonzero, then .POADR contains the first address in the range, .POADE contains the last address in the range, and the range includes all the addresses between them.
- o If both .POADR and .POADE are zero, the range is all of memory.
- o If .POADE is zero and .POADR is not, the range begins at .POADR and includes all higher addresses in the rest of memory.
- o If .POADE is not zero, and .POADR is larger than .POADE, an error results.

You can use the following function codes in AC1.

| Code | Symbol | Function |
|------|--------|--|
| 0 | .POGET | For the process specified in word .POPHD of the argument block, this function returns all PDVA's within the range of addresses specified in words .POADR and .POADE of the argument block. |
| 1 | .POADD | This function adds the PDVA's specified in the data block to the system's data base for the specified process. The PDVA's must be in ascending order within the data block. |
| 2 | .POREM | This function removes a set of PDVA's from the system's data base for the specified process. The PDVA's removed are the ones within the range of addresses specified in words .POADR and .POADE of the argument block. |
| 3 | .PONAM | This function returns the ASCIZ name of a program in memory. Word .POADR of the argument block must contain a valid PDVA for the specified process. The name returned is the one to which word .PVNAM of the PDV points. |
| 4 | .POVER | This function returns the version of a program in memory. Word .POADR must contain a valid PDVA for the specified process. The version returned is the one that word .PVVER of the PDV contains. |
| 5 | .POLOC | For the specified process, this function returns all the PDVA's of PDV's for the specified program. |

TOPS-20 MONITOR CALLS
(PDVOP%)

The byte pointer in AC3 points to the program name.

This call generates an illegal instruction interrupt on the error conditions below.

PVDOP% ERROR MNEMONICS:

ARGX06: Invalid page number
MONX02: Insufficient system resources (JSB full)
PDVX01: Address in .POADE must be as large as address in .POADR
PDVX02: Addresses in .PODAT block must be in strict ascending order
PDVX03: Address in .POADR must be a program data vector address
FDKHX8: Illegal to manipulate an execute-only process

Transfers a block of words from the monitor's address space to the user's address space. The desired monitor words must exist on pages that have read access. This monitor call is used to obtain data from the monitor for maintenance and test purposes and should be executed only when GETAB information is not available.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Word count in the left half, and first virtual address of the monitor in the right half

AC2: First user address

RETURNS +1: Failure, error code in AC1

+2: Success, the desired words are transferred.

PEEK ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required
PEEKX2: Read access failure on monitor page

TOPS-20 MONITOR CALLS
(PLOCK)

Acquires physical memory and places a designated section of the process's address space in memory. Allows the process to specify the memory pages to be used, or permits the system to select the pages.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Address of first page if acquiring (locking) or -1 if unlocking

AC2: Process handle (currently .FHSLF only) in the left half and number of first page in the right half

AC3: Control flags in the left half and repeat count in the right half. The control flags are:

B0 (LK%CNT) Right half of AC3 contains a count of the number of pages to lock.

B1 (LK%PHY) Value in AC1 is the first page desired. If this bit is off and AC1 is not -1, the system selects pages.

B2 (LK%NCH) Pages will not be cached.

B3 (LK%AOL) Off-line pages are to be locked.

B4 (LK%EPN) Page number is absolute and not relative to a section.

RETURNS +1: Always

If the PLOCK call is unable to honor any one of the requests to unlock any one of the pages specified by the repeat count, it will unlock all of the others.

A page that was locked with the PLOCK call may be unmapped. (See the PMAP call.) This will unlock the process's page and return the now unlocked physical page to its previous state.

The page selected by the user must be capable of being placed off-line for the PLOCK call to acquire it.

The use of PLOCK to lock many pages at a time can cause a system crash on a loaded system. The proper method is to lock pages only in small block allocations (2-10 pages at a time), rather than use several hundred page block allocations. Alternatively, the user can check the change in system free pages (NRPLQ) over a period of time and not lock more than one-half the number of freed pages in a recent interval.

TOPS-20 MONITOR CALLS
(PLOCK)

Generates an illegal instruction interrupt on error conditions below.

PLOCK ERROR MNEMONICS:

ARGX22: Invalid flag
ARGX24 invalid count

Maps one or more complete pages from a file to a process (for input), from a process to a file (for output), or from one process to another process. Also unmaps pages from a process and deletes pages from a file. Each of the five uses of PMAP is described below.

Case I: Mapping File Pages to a Process

This use of the PMAP call does not actually transfer any data; it simply changes the contents of the process's page map. When changes are made to the page in the process, the changes will also be reflected in the page in the file, if write access has been specified for the file.

ACCEPTS IN AC1: JFN of the file in the left half, and the page number in the file in the right half. This AC contains the source.

AC2: Process handle in the left half, and the page number in the process in the right half. This AC contains the destination.

AC3: Access bits,,repetition count

B0(PM%CNT) A count is in the right half of AC3. This count specifies the number of sequential pages to be mapped. If this bit is not set, one page is mapped.

B2(PM%RD) Permit read access to the page.

B3(PM%WR) Permit write access to the page.

B4(PM%EX) Reserved for future use.
The symbol PM%RWX can be used to set B2-4.

TOPS-20 MONITOR CALLS
(PMAP)

- B5(PM%PLD) Preload the page being mapped (move the page immediately instead of waiting until it is referenced).
- B9(PM%CPY) Create a private copy of the page when it is written into (copy-on-write). If the page is mapped between two processes (Case III below), both processes will receive a private copy of the page.
- B10(PM%EPN) The right half of AC2 contains an extended process page number. If the section containing the page does not exist, an illegal instruction trap is generated.
- B11(PM%ABT) Unmap a page and throw its changed contents away. This bit is significant only when unmapping process pages that were mapped from a file (see case IV below) and OF%DUD is set in the OPENF.

Normally, if a page is unmapped and has been changed since the last time the monitor updated the associated file page, the monitor will remove the page from the process and place it on a queue in order to update the file page. PM%ABT allows the page to be unmapped, but prevents the monitor from placing the page on the update queue.

This feature is useful in the case of erroneous data written to a mapped page of a file open for simultaneous access. In this case, it is important that the erroneous page be discarded and not be used to update the file page. Another application is to allow processes in separate jobs to communicate by sharing a file page (and reading/writing the page) and avoid the overhead of the monitor periodically updating the page.

- B18-35 Number of pages to be mapped if
(PM%RPT) B0(PM%CNT) is set.

RETURNS +1: Always

This use of PMAP changes the map of the process such that addresses in the process page specified by the right half of AC2 actually refer to

TOPS-20 MONITOR CALLS
(PMAP)

the file page specified by the right half of AC1. The present contents of the process page are removed. If the page in the file is currently nonexistent, it will be created when it is written (when the corresponding page in the process is written). If the process page is in a nonexistent section, an illegal instruction trap is generated.

This use of PMAP is legal only if the file is opened for at least read access. The access bits specified in the PMAP call are ANDed with the access that was specified when the file was opened. However, copy-on-write is always granted, regardless of the file's access. The access granted is placed in the process's map. The file cannot be closed while any of its pages are mapped into any process. Thus, before the file is closed, pages must be unmapped from each process by a PMAP call with -1 in AC1 (see below).

Case II Mapping Process Pages to a File

This use of the PMAP call actually transfers data by moving the contents of the specified page in the process to the specified page in the file. The process's map for that page becomes empty.

ACCEPTS IN AC1: Process handle in the left half, and the page number within the process in the right half. This AC contains the source.

AC2: JFN of the file in the left half, and the page number within the file in the right half. This AC contains the destination.

AC3: Access bits and repetition count. (Refer to Case I.)

RETURNS +1: Always

The process page and the file page must be private pages. The ownership of the process page is transferred to the file page. The present contents of the page in the file is deleted.

The access granted to the file page is determined by ANDing the access specified in the PMAP call with the access specified when the file was opened. This function does not update the file's byte size or the end-of-file pointer in the file's FDB. Failure to update these items in the FDB can prevent the reading of the file by sequential I/O calls such as BIN and BOUT.

To update the file's FDB after using this PMAP function, do the following:

1. Use the CLOSF call with the CO%NRJ bit set to close the file but keep the JFN.

TOPS-20 MONITOR CALLS
(PMAP)

2. Use the CHFDB call to update the end-of-file pointer and, if necessary, the byte size in the file's FDB.
3. Use the RLJFN call to release the JFN.

(See Section 2.2.8 for the format of the FDB fields.)

Case III Mapping One Process's Pages to Another Process

This use of the PMAP call normally does not transfer any data; it simply changes the contents of the page maps of the processes. When changes are made to the page in one process, the changes will also be reflected in the corresponding page in the other process.

ACCEPTS IN AC1: Process handle in the left half, and the page number in the process in the right half. This AC contains the source.

AC2: A second process handle in the left half, and page number in that process in the right half. This AC contains the destination.

AC3: Access bits and repetition count. (Refer to Case I.)

RETURNS +1: Always

This use of PMAP changes the map of the destination process such that addresses in the page specified by the right half of AC2 actually refer to the page in the source process specified by the right half of AC1. The present contents of the destination page are deleted.

The access granted to the destination page is determined by the access specified in the PMAP call. If the destination page is in a nonexistent section, the monitor generates an illegal instruction trap.

Case IV Unmapping Pages In a Process

As stated previously, a file cannot be closed if any of its pages are mapped in any process.

ACCEPTS IN AC1: -1

AC2: Process handle in the left half, and page number within the process in the right half

AC3: Access bits,,repetition count

B0(PM%CNT) RH contains the number of pages to delete

TOPS-20 MONITOR CALLS
(PMAP)

B10(PM%EPN) Extended page number (18 bits)

B11(PM%ABT) Unmap page and abort contents

B18-35 Number of pages to remove from process
(PM%RPT)

Only these bits have meaning on this
call. All others are ignored.

This format of the PMAP call removes the pages indicated in AC2 from the process.

A page that was locked with the PLOCK call may be unmapped. Doing so will unlock the process's page and return the now unlocked physical page to its previous state.

Case V Deleting One or More Pages from a File

Deletes one or more pages from a file on disk and does not affect the address space of any process.

ACCEPTS IN AC1: -1

AC2: JFN of the file in the left half and page number
within the file in the right half.

AC3: B0(PM%CNT) Indicates that the right half contains
the number of pages to delete.

B18-35 Number of pages to delete from file
(PM%RPT)

Illegal PMAP calls

The PMAP call is illegal if:

1. Both AC1 and AC2 designate files.
2. Both AC1 and AC2 are 0.
3. The PMAP call designates a file with write-only access.
4. The PMAP call designates a file with append-only access.
5. The source and/or the destination designates an execute-only process and the process is not self (.FHSLF).

Can cause several software interrupts on certain file conditions.

TOPS-20 MONITOR CALLS
(PMAP)

Generates an illegal instruction interrupt on error conditions below.

PMAP ERROR MNEMONICS:

ARGX06: Invalid page number
CFRKX3: Insufficient system resources
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DESX7: Illegal use of parse-only JFN or output wildcard-designators
FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX7: Process page cannot exceed 777
FRKHX8: Illegal to manipulate an execute-only process
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged
LNGFX1: Page table does not exist and file not open for write
PMAPX1: Invalid access requested
PMAPX2: Invalid use of PMAP
PMAPX3: Illegal to move shared page into file
PMAPX4: Illegal to move file page into process
PMAPX5: Illegal to move special page into file
PMAPX6: Disk quota exceeded
PMAPX7: Illegal to map file on dismounted structure
PMAPX8: Indirect page map loop detected

WARNING: This JSYS can cause a system crash. Use with extreme caution.

NOTE

This JSYS is primarily intended for system use. The information returned may change in a future release.

Controls physical memory. This call allows a privileged program to add or remove most pages of physical memory and to control use of cache memory.

RESTRICTIONS: Requires WHEEL, MAINTENANCE or OPERATOR capability enabled.

TOPS-20 MONITOR CALLS
(PMCTL)

ACCEPTS IN AC1: Function code

AC2: Length of the argument block

AC3: Address of the argument block

RETURNS +1: Always

The defined functions and their argument blocks are as follows:

| Function | Symbol | Meaning |
|----------|--------|--|
| 0 | .MCRCE | Return the status of cache memory. The status is returned in word .MCCST of the argument block. Argument Block 0 .MCCST If B35(MC%CEN) is on, the cache is enabled. |
| 1 | .MCSCE | Set the status of cache memory. Argument Block 0 .MCCST Enable the cache if B35(MC%CEN) is on. |
| 2 | .MCRPS | Return the status of the given page(s). The number of the page is given in word .MCPN, and its status is returned in word .MCPST. Argument Block 0 .MCPN Negative count in the left half; number of physical page in the right half 1 .MCPST Returned page status. The status is represented by one of the following values: 0 .MCPSA Page is available for normal use. 1 .MCPSS Page is in a transition state. 2 .MCPSO Page is off line because it is nonexistent. |

TOPS-20 MONITOR CALLS
(PMCTL)

Nonexistent memory
is marked as off
line at system
startup.

- 3 .MCPSE Page is off line
because the monitor
detected an error.
- 3 .MCSPS Set the status of the given page. The number
of the page is given in word .MCPN, and the
status value is given in word .MCPST.
- Argument Block
- 0 .MCPN Number of physical page.
- 1 .MCPST Status for page. The status is
represented by one of the
following values:
- 0 .MCPSA Mark page available
for normal use.
- 1 .MCPSS Mark page in
transition
- 2 .MCPSO Mark page off line
because it does not
exist.
- 3 .MCPSE Mark page off line
because it has an
error.
- 4 .MCRME Collect information about MOS memory errors.
Store the information in block addressed by
AC3 and update AC2 on return.

A list of those pages that PMCTL cannot acquire follows:

1. the EPT
2. the monitor's UPT
3. any page containing a CST0 entry
4. any page containing an SPT entry
5. the page containing MMAP

TOPS-20 MONITOR CALLS
(PMCTL)

6. any page belonging to the resident free space pool
7. any page containing a monitor page table

In certain specialized monitors, for example TOPS-20AN, there are additional pages that cannot be acquired. An estimate of the size of these areas follows:

| | |
|--------------------------|---|
| CST0 | one word for every page of memory supported (two to four pages) |
| SPT | four pages |
| MMAP | one page |
| Resident Free Space Pool | two pages minimum |

Generates an illegal instruction interrupt on error conditions below.

PMCTL ERROR MNEMONICS:

| | |
|---------|---|
| CAPX2: | WHEEL, OPERATOR, or MAINTENANCE capability required |
| PMCLX1: | Invalid page state or state transition |
| PMCLX2: | Requested physical page is unavailable |
| PMCLX3: | Requested physical page contains errors |
| ARGX02: | Invalid function |
| ARGX06: | Invalid page number |

Translates a project-programmer number (a TOPS-10 36-bit directory designator) to its corresponding TOPS-20 string. The string consists of the structure name and a colon followed by the directory name enclosed in brackets. This monitor call and the STPPN monitor call should appear only in programs that require translations of project-programmer numbers. Both calls are temporary calls and may not be defined in future releases.

ACCEPTS IN AC1: Destination designator

AC2: Project-programmer number (36 bits)

AC3: Byte pointer to structure name string for which the given project-programmer number applies.

RETURNS +1: Always, with string written to destination, with updated byte pointer, if pertinent, in AC1

TOPS-20 MONITOR CALLS
(PPNST)

If the structure name string is a logical name, then the first structure appearing in the logical name definition is used.

Generates an illegal instruction interrupt on error conditions below.

PPNST ERROR MNEMONICS:

PPNX1: Invalid PPN
PPNX2: Structure is not mounted
GJFX22: Insufficient system resources (Job Storage Block full)
STDVX1: No such device
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
DELFX6: Internal format of directory is incorrect
DIRX1: Invalid directory number
DIRX2: Insufficient system resources
DIRX3: Internal format of directory is incorrect
STRX01: Structure is not mounted
STRX06: No such user number
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Returns or sets up an argument block for the specified process. The monitor stores the argument block in process storage block for this process.

This call is useful for running a program whenever another program halts. Examples are running a compiler or re-executing the last compile-class command each time you exit an editor.

This call uses the 200-word process storage block associated with each process. User programs can only access this memory by means of the PRARG monitor call. A process and all of its superior processes can access the process storage block of a given process. Furthermore, data associated with many different programs can be stored a given process storage block.

ACCEPTS IN AC1: Function code in the left half, and a process handle in the right half

TOPS-20 MONITOR CALLS
(PRARG)

AC2: Address of argument block

AC3: Length of argument block

RETURNS +1: Always, with the number of words of data in the
 returned argument block in AC3

The codes for the functions are as follows:

- | | | |
|---|--------|--|
| 1 | .PRARD | Return the arguments beginning at the address specified in AC2 |
| 2 | .PRAST | Set the arguments using the argument block at the address specified in AC2 |

The PRARG argument block has the following format:

| Offset | Meaning |
|--------|--|
| 0 | Number of argument blocks |
| 1 | Relative address (from the start of this block) of the first argument list |
| 2 | Relative address of the second argument list . . . |
| N | Relative address of the Nth argument list |

The argument list format is the following:

| Word | Meaning |
|------|--|
| 0 | Number of argument lists (must be 1) |
| 1 | Entry type in the left half (must be 400740), and the address, relative to the start of the argument block, of the argument list in the right half (usually 2, but other relative addresses are allowed) |

The argument list contains an ASCII string that is the name of the program to run; or the list contains a zero, which means that the last compile-class command is to be re-executed.

Generates an illegal instruction interrupt on error conditions below.

PRARG ERROR MNEMONICS:

- | | |
|--------|---|
| PRAX1: | Invalid PRARG function code |
| PRAX2: | No room in monitor data base for argument block |
| PRAX3: | PRARG argument block too large |

TOPS-20 MONITOR CALLS
(PSOUT)

Outputs a string sequentially to the primary output designator.

ACCEPTS IN AC1: Byte pointer to an ASCIZ string in the caller's address space

RETURNS +1: Always, with updated byte pointer in AC1

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

PSOUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX2: File is not open for writing
IOX5: Device or data error
IOX6: Illegal to write beyond absolute end of file
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Provides a mechanism for communicating with the operator as well as a mechanism for initiating queue requests.

Two essential pieces of information are needed to use QUEUE%:

- o Function type - Queueing request, write-to-operator
- o Set of argument blocks appropriate for the function type

QUEUE% provides two classes of functions. One class, the actual queuing functions, causes a job request to be presented to QUASAR for processing, similar to submit and print commands. The other class enables limited communications with the operator, providing the same functions as the PLEASE program.

ACCEPTS IN AC1: Length of argument block

TOPS-20 MONITOR CALLS
(QUEUE%)

AC2: Address of argument block

RETURNS +1: Always

The user program builds the main argument block containing header information and various other argument blocks that declare attributes of the request. The format of the main argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|-------------------------|
| 0 | .QFNC | B0-B7(QF%FLG) Flag bits |

B0(QU%NRS) No response (don't wait)
In addition to performing the requested function, QUEUE% returns a response unless a flag is set explicitly declining a response. For the queuing functions, the response is an ASCII string indicating the job has been accepted (same as the acknowledgement line provided in response to a queue request in the EXEC). The response has a slightly different meaning depending on use of the write-to-operator functions, as described below.

B1(QU%DBG) Use system-wide debugging PID

B8-B17 (QF%RSP) Length of response block (1 page maximum; see QU%NRS)

B18-B35(QF%FNC) Function code

Queuing Functions -- Queuing functions perform tasks normally accomplished with PRINT and SUBMIT commands. For these functions, a file descriptor argument is required before any other argument blocks. Any number of other argument blocks may be included after the file specification to declare various attributes of the request. These arguments are similar to the switches associated with those commands.

| | | |
|---|--------|------------------|
| 1 | .QUPRT | Print file |
| 2 | .QUCDP | Punch cards |
| 3 | .QUPTP | Punch paper tape |
| 4 | .QUPLT | Plot file |
| 5 | .QUBAT | Submit batch job |

Write-to-Operator -- The write-to-operator functions perform the same functions normally associated with use of the PLEASE program. The response to this type of function depends on the function. For a write-to-operator without reply,

TOPS-20 MONITOR CALLS
(QUEUE%)

the acknowledgement indicates that the message has been received. For a write-to-operator with reply, the process will remain blocked until the operator responds to the message which should be in the form of a request. In this case, the response is the actual reply.

| | | |
|-------|----------|------------------------------|
| 12 | .QUWTO | Write-to-operator |
| 13 | .QUWTR | Write-to-operator with reply |
| 14-15 | Reserved | |
| 16 | .QUCUS | Use custom application PID |

1 .QURSP Address of response block

2 .QUARG First of n contiguous attribute argument blocks. These specify the function parameters. Each two-word argument block has the following general format:

| Word | Symbol | Contents | | | | | | | | | | | | |
|-------|--------|--|-----|--------|---------|---|--------|---|------|--------|--|-------|--------|--|
| 0 | .QATYP | First word of argument block | | | | | | | | | | | | |
| | | <table border="0"> <thead> <tr> <th>Bit</th> <th>Symbol</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>QA%IMM</td> <td>If set, implies immediate argument value. Argument value is contained in word .QADAT.</td> </tr> <tr> <td>9-17</td> <td>QA%LEN</td> <td>Length of argument value (1 if QA%IMM is set).</td> </tr> <tr> <td>18-35</td> <td>QA%TYP</td> <td>Argument code (see individual argument block descriptions for possible codes).</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | 0 | QA%IMM | If set, implies immediate argument value. Argument value is contained in word .QADAT. | 9-17 | QA%LEN | Length of argument value (1 if QA%IMM is set). | 18-35 | QA%TYP | Argument code (see individual argument block descriptions for possible codes). |
| Bit | Symbol | Meaning | | | | | | | | | | | | |
| 0 | QA%IMM | If set, implies immediate argument value. Argument value is contained in word .QADAT. | | | | | | | | | | | | |
| 9-17 | QA%LEN | Length of argument value (1 if QA%IMM is set). | | | | | | | | | | | | |
| 18-35 | QA%TYP | Argument code (see individual argument block descriptions for possible codes). | | | | | | | | | | | | |
| 1 | .QADAT | Address of argument or argument value if QA%IMM is set. | | | | | | | | | | | | |

The following section describes each of the attribute argument blocks.

| Code | Symbol | Meaning/Arguments |
|------|--------|--|
| 10 | .QBFIL | This argument block (file specification) is required for all queuing functions. For a PRINT job, it indicates the file to be printed. For a BATCH job, it indicates the control file to be |

TOPS-20 MONITOR CALLS
(QUEUE%)

used for the batch job. The file descriptor argument block must be specified before any other attributes. Argument: ASCII text (filename as ASCII string).

- | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|--------|--|---|--------|--------------------|---|--------|--------------------|---|--------|-------------------------------------|---|--------|-----------------|---|--------|--------------|---|--------|--------|---|--------|-------|----|--------|-------------|
| 11 | .QBCOP | Indicates the number of copies to be generated. For use exclusively with output (PRINT) requests. Argument: Number of copies. | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | .QBFRM | Indicates the form to be used for the output. Form indicates paper type as well as some of the print characteristics such as width and length of a printed page. For use with output queue requests, PRINT. Argument: forms name in SIXBIT. | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | .QBFMT | Describes the format of the file. Using this information the printer spooler can correctly interpret the data in the file for printing. Arguments: <table border="0" style="margin-left: 40px;"> <tr><td>1</td><td>.QBFAS</td><td>ASCII</td></tr> <tr><td>2</td><td>.QBFFR</td><td>FORTTRAN</td></tr> <tr><td>3</td><td>.QBFCB</td><td>COBOL</td></tr> <tr><td>4</td><td>.QBFAI</td><td>Augmented Image</td></tr> <tr><td>5</td><td>.QBFSA</td><td>Stream ASCII</td></tr> <tr><td>6</td><td>.QNF11</td><td>Eleven</td></tr> <tr><td>7</td><td>.QBFIM</td><td>Image</td></tr> <tr><td>10</td><td>.QBF8B</td><td>8-bit ASCII</td></tr> </table> | 1 | .QBFAS | ASCII | 2 | .QBFFR | FORTTRAN | 3 | .QBFCB | COBOL | 4 | .QBFAI | Augmented Image | 5 | .QBFSA | Stream ASCII | 6 | .QNF11 | Eleven | 7 | .QBFIM | Image | 10 | .QBF8B | 8-bit ASCII |
| 1 | .QBFAS | ASCII | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | .QBFFR | FORTTRAN | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | .QBFCB | COBOL | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | .QBFAI | Augmented Image | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | .QBFSA | Stream ASCII | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | .QNF11 | Eleven | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | .QBFIM | Image | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | .QBF8B | 8-bit ASCII | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | .QBODP | Indicates whether certain files associated with this request are to be deleted or kept (preserved) upon completion of the job. For use with any of the queuing functions. In a PRINT job, the printed files are deleted or preserved. In a BATCH job, it is the control file that is preserved or deleted with this parameter. Argument: 0 to preserve, 1 to delete. | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | .QBUNT | Indicates the unit (object) number and characteristics of the object for processing the job. For use with any of the queuing functions. The unit number indicates the stream number in the case of a BATCH job. The physical characteristics are only applicable to PRINT requests. Arguments: <table border="0" style="margin-left: 40px;"> <tr><td>1</td><td>.QBULC</td><td>Lower case printer</td></tr> <tr><td>2</td><td>.QBUUC</td><td>Upper case printer</td></tr> <tr><td>3</td><td>.QBUPH</td><td>Physical unit number provided in LH</td></tr> <tr><td>4</td><td>.QBUGN</td><td>Generic device</td></tr> </table> | 1 | .QBULC | Lower case printer | 2 | .QBUUC | Upper case printer | 3 | .QBUPH | Physical unit number provided in LH | 4 | .QBUGN | Generic device | | | | | | | | | | | | |
| 1 | .QBULC | Lower case printer | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | .QBUUC | Upper case printer | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | .QBUPH | Physical unit number provided in LH | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | .QBUGN | Generic device | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | .QBAFT | Allows a job to be started at some future time. | | | | | | | | | | | | | | | | | | | | | | | | |

TOPS-20 MONITOR CALLS
(QUEUE%)

For use with any queuing request. Argument:
Date/time in UDT format.

- | | | |
|----|--------|--|
| 17 | .QBLIM | Limits the amount of resources allocated to this job. Also has a secondary use as an attribute that is considered in the scheduling of jobs. For use with any of the queuing functions. For PRINT jobs, it indicates the maximum number of pages to be printed. For BATCH jobs, it indicates the time limit for the job. Argument: Limit of job as number. |
| 20 | .QBUNQ | Enables the user to allow/disallow the simultaneous running of multiple batch jobs. For use with BATCH requests only. Arguments: 1 .QBUNO No 2 .QBUYE Yes |
| 21 | .QBRES | Allows the job to be restarted after a system failure. For use with BATCH requests only. Arguments: 1 .QBRNO No 2 .QBRYE Yes |
| 22 | .QBLOG | Indicates the conditions upon which a log file is to be generated. Appropriate for use with BATCH jobs only. Arguments: 1 .QBLNL No log file is to be generated. 2 .QBLLG Always generate a log file. 3 .QBLLE Generate a log file only if an error occurs. |
| 23 | .QBACT | Indicates the account to be charged for job execution. For use with all queuing functions. Argument: ASCIZ text (account as ASCII string). |
| 24 | | Reserved for DIGITAL. |
| 25 | .QBNOD | Associates a node with the request. Interpretation depends on the context. For a write-to-operator, this indicates that the message is destined for operators only on the node specified. For PRINT requests, it indicates the node on which the printing is to occur. Argument: Node name in SIXBIT. |
| 26 | .QBNAM | 6-bit user name (maximum 12 characters). |

TOPS-20 MONITOR CALLS
(QUEUE%)

- 27 .QBOID Identifies the user by his logged in directory number. For use with any queuing request. Argument: user number.
- 30 .QBNOT Enables the requestor to be notified upon completion of the job. For use with any queuing request. Arguments: 0 if no notify, 1 (.QBNTY) to notify.
- 31 .QBBLT Indicates how the log file should be created/disposed. Appropriate for use with BATCH jobs only. Arguments:
- 1 .QBBND Append log file for this job to currently existing log file.
- 2 .QBBDE Supersede the currently existing log file.
- 3 .QBBSP Spool the log file on completion of the job.
- 32 .QBJBN Sets a jobname other than the default (generated from the first 6 characters of the filename in the queue request). For use with any of the queuing functions. Argument: Jobname in SIXBIT (from 1 to 6 SIXBIT characters). This jobname can be used for modifications to the request with the MODIFY and CANCEL commands.
- 33 .QBCDI 36-bit directory number.
- 34 .QBNTY Allows up to 12 SIXBIT characters to be associated with a queuing request as a note. For use with output (PRINT) requests. Argument: SIXBIT text.
- 35 .QBBGN Specifies the beginning of processing of the job. For use with any of the queuing functions. Depending on the queuing function, the attribute can have different meanings. For PRINT jobs, it indicates the number of the page on which printing is to begin. For BATCH jobs, it indicates processing is to start at the line number indicated. Argument: Number indicating where to begin.
- 36 .QBPRI Allows the user to specify the priority of the job for scheduling purposes only. For use with any queuing requests. Argument: Number 0<#<63 indicating priority. There are some restrictions on which priorities may be selected by nonprivileged users.

TOPS-20 MONITOR CALLS
(QUEUE%)

- 37 .QBVSN Volume set name in ASCIZ.
- 40 .QBMSG Used to send a text message from one GALAXY component to another, generally for display purposes. For use with write-to-operator messages (with or without reply). Argument: ASCIZ text (text containing message).
- 41 .QBTYP Used to send a text message from one GALAXY component to another, generally for display purposes. The sender of this type of message is checked for privileges, since it replaces the header information of the OPR display message. For use with write-to-operator messages (with or without reply). Argument: ASCIZ text (text containing message).
- 53 .QBDTY Indicates the type of display message. For use with write-to-operator messages (with or without reply). Arguments:
- 1 .QBCHK Indicates BUGCHK display (monitor use only).
- 2 .QBINF Indicates BUGINF display (monitor use only).
- 3 .QBSYS Indicates SYSTEM messages (monitor use only).
- 4 .QBEVT Indicates DECnet event messages.
- 5 .QBDLK Indicates DECnet link messages.
- 54 .QBSNA Sets the SNA parameters block. Arguments:
- 0 QU%TABS Preserve tabs in file.
- 1 QU%NXL Do not translate data.
- 2-35 QU%RCL Record length
- 55 .QBDFG Display flags (used with write-to-operator). Arguments:
- 0 QU%SJI Suppress job information.
- 1 QU%NFO Do not format display.
- 2 QU%NFA Do not include dashes in type display.

QUEUE% ERROR MNEMONICS:

- QUEUX1: Illegal argument list passed to QUEUE%
- QUEUX2: Invalid function
- QUEUX3: Fatal error returned from application
- QUEUX4: Invalid message returned from ORION
- QUEUX5: Insufficient system resources (Job Storage Block full)

TOPS-20 MONITOR CALLS
(QUEUE%)

QUEUX6: Illegal response length
QUEUX7: Argument block too small

Translates the given directory string to its corresponding 36-bit directory number.

A directory string contains a structure name and a directory name. The structure name must be followed by a colon, and the directory name must be enclosed in either square brackets or angle brackets. No spaces can appear between the structure name and the directory name. Here is an example of a directory string:

PS:<SMITH>

Recognition cannot be used on the structure name. If the structure name is omitted from the string, the user's connected structure is used. Wildcards cannot be used in the structure name field.

Recognition can be used on the directory name field. Recognition can also be used on part of the directory name field, so that a user can employ recognition when typing the name of a subdirectory. When recognition is used on the directory name field, and the directory name is not ambiguous, the closing bracket is not required.

Wildcards can be used in the directory name field. Repeated RCDIR calls can be executed to obtain the numbers of the directories whose names match the given directory string. After the first call, each subsequent RCDIR call returns the number of the next directory that matches the directory string.

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Flag bits in the left half

AC2: Byte pointer to ASCIZ string to be translated, a JFN, a 36-bit user number, or a 36-bit directory number (given for the purpose of checking its validity)

AC3: 36-bit directory number (given when stepping to the next directory in a group of directories)

TOPS-20 MONITOR CALLS
(RCDIR)

RETURNS +1: Always, with

AC1 containing flag bits in the left half

AC2 containing an updated byte pointer (if a pointer was supplied as the argument). If recognition was used, this pointer reflects the remainder of the string that was appended to the original string.

AC3 containing a 36-bit directory number if execution of the call was successful

The flag bits supplied in the left half of AC1 are as follows:

B14(RC%PAR) Allow partial recognition on the directory name. If the name given matches more than one directory, bit RC%AMB is set on return and the string is updated to reflect the unique portion of the directory name.

If bit RC%PAR is not set, the name given matches more than one directory, and recognition is being used, then bit RC%AMB is set on return, but the string is not updated.

B15(RC%STP) Step to the next directory in the group and return the number of that directory. AC1 must have bit RC%AWL set. AC2 must contain a pointer to a string that contains wildcard characters in the directory name field. AC3 must contain a directory number.

B16(RC%AWL) Allow the directory name to contain wildcard characters. The directory name must include its terminating bracket. No recognition is performed on a directory name that contains wildcard characters.

This bit must be set if bit RC%STP is also set.

B17(RC%EMO) Match the given string exactly. When both the RC%PAR and RC%EMO bits are on, recognition is not used on the string, and the string is matched exactly.

If this bit is off, recognition is used on the string.

The flag bits returned in the left half of AC1 are as follows:

On success

B0(RC%DIR) Directory can be used only by connecting to it. (It is a files-only directory.)

If this bit is off, the user can also login to (if the

TOPS-20 MONITOR CALLS
(RCDIR)

directory is on the public structure) or access this directory.

- B1(RC%ANA) Obsolete
- B2(RC%RLM) All messages from <SYSTEM>MAIL.TXT are repeated every time the user logs in. If this bit is off, messages are printed only once.
- B6(RC%WLD) The directory name given contained wildcard characters.

On failure

- B3(RC%NOM) No match was found for the string given. This bit is returned if either 1) bit RC%EMO was on in the call, and a string was given that matched more than one directory; or 2) the syntax of the fields in the string is correct, but the structure is not mounted, or the directory does not exist.
- B4(RC%AMB) The argument given was ambiguous. This bit is returned if bit RC%EMO was off, and if the string given either matched more than one directory, or did not include the beginning bracket of the directory name field.
- B5(RC%NMD) There are no more directories in the group of directories. This bit is returned if RC%STP was on and the numbers of all the directories in the group have been returned.

The RCDIR monitor call can be used in one of two ways. The simpler way is to translate a directory string to its corresponding 36-bit directory number. The string can be either recognized, or matched exactly.

The second way of using the RCDIR call is to provide a directory string that corresponds to more than one directory, and then use repeated RCDIR calls to step through all the directories matching the given string. Each call obtains the number of the next directory that matches the given string. When no more directories match the string, the RC%NMD bit is set on the call's return.

When obtaining a single directory number, RCDIR can accept a JFN, a 36-bit user number, or a directory number. When a JFN is supplied as an argument, the number returned is that of the directory containing the file associated with the JFN. When a user number is supplied as an argument, the number returned is the logged-in directory for that user. When a directory number is supplied, the RCDIR call checks the number's validity. If the number is valid, the RCDIR call is successful, and this same number is returned.

TOPS-20 MONITOR CALLS
(RCDIR)

When obtaining several directory numbers, RCDIR requires AC2 to contain a pointer to a directory string that contains wildcard characters. If the string does not contain wildcards, or if any thing other than a string pointer is given in AC2, the stepping function is not performed, and the call returns with the RC%NMD bit set.

Furthermore, the first RCDIR call executed must have bit RC%AWL set in AC1, and the pointer to the string in AC2. If execution of the call is successful, AC3 contains the number of the directory corresponding to the first directory that matches the given directory string. For example, if the string given is <SMITH*> and the call is successful, the number returned corresponds to <SMITH>.

Subsequent RCDIR calls must set bits RC%STP and RC%AWL in AC1, reset the pointer in AC2 (because it is updated on a successful RCDIR call), and leave in AC3 the directory number returned from the previous RCDIR call. The directory number in AC3 is accepted only if RC%STP is set in AC1, and a pointer to a string containing wildcard characters is given in AC2.

On successful execution of each subsequent RCDIR call, the number returned in AC3 corresponds to the next directory in the group. When the number of the last directory in the group has been returned, a subsequent RCDIR call sets bit RC%NMD in AC1; the content of AC3 is indeterminate.

The RCUSR monitor call can be used to translate a user name string to its corresponding user number. The DIRST monitor call can be used to translate either a directory number or a user number to its corresponding string.

Generates an illegal instruction interrupt on error conditions below.

RCDIR ERROR MNEMONICS:

RCDIX1: Insufficient system resources
RCDIX2: Invalid directory specification
RCDIX3: Invalid structure name
RCDIX4: Monitor internal error
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DESX8: File is not on disk
DESX10: Structure is dismounted
STRX01: Structure is not mounted
STRX10: Structure is offline

TOPS-20 MONITOR CALLS
(RCM)

Returns the word mask of the activated interrupt channels for the specified process. (See Section 2.6.1 and the AIC and DIC calls for information on activating and deactivating software interrupt channels.)

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with 36-bit word in AC1, with bit n on, meaning channel n is activated

Generates an illegal instruction interrupt on error conditions below.

RCM ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Translates the given user name string to its corresponding 36-bit user number. The user name string consists of the user's name without any punctuation. The string must be associated with a directory on the public structure (usually called PS:) that is not a files-only directory.

Recognition can be used on the string. In addition, the string can contain wildcard characters.

ACCEPTS IN AC1: Flag bits in the left half

AC2: Byte pointer to ASCII string to be translated

AC3: 36-bit user number (given when stepping to the next user name in a group)

RETURNS +1: Always, with

AC1 containing flag bits in the left half

AC2 containing an updated byte pointer. If recognition was used, this pointer reflects the remainder of the string that is appended to the original string.

TOPS-20 MONITOR CALLS
(RCUSR)

AC3 containing a 36-bit user number if execution of the call was successful. An example of a user number is: 500000,,261.

The flag bits supplied in the left half of AC1 are as follows. For additional information on these bits, see the RCDIR monitor call description.

- B14(RC%PAR) Allow partial recognition on the user name string.
- B15(RC%STP) Step to the next user name in the group.
- B16(RC%AWL) Allow the user name to contain wildcard characters.
- B17(RC%EMO) Match the given string exactly.

The flag bits returned in the left half of AC1 are as follows. For additional information on these bits, see the RCDIR monitor call description.

On success

- B1(RC%ANA) Obsolete
- B2(RC%RLM) User sees all messages from <SYSTEM>MAIL.TXT every time he logs in. If this bit is off, the user sees the messages only once.
- B6(RC%WLD) The user name given contained wildcard characters.

On failure

- B3(RC%NOM) No match was found for the string given. This bit will be on if the string given refers to a files-only directory, if there is no directory on PS: that is associated with the user name string, or bit RC%EMO was on in the call and a string was given that matched more than one user.
- B4(RC%AMB) The string given was ambiguous because it matched more than one user.
- B5(RC%NMD) There are no more user names in the group.

The RCDIR monitor call can be used to translate a directory string to its corresponding directory number. The DIRST monitor call can be used to translate either a user number or a directory number to its corresponding string.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(RCUSR)

RCUSR ERROR MNEMONICS:

RCUSX1: Insufficient system resources
RCDIX4: Monitor internal error
STRX07: Invalid user number
STRX08: Invalid user name

Retrieves a message from the TCP/IP special message queue. The queue must have been previously assigned with the ASNSQ% JSYS.

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: B0 If on, the user will receive a 96-bit leader. If off, the user will receive a 32-bit leader.
B1 If on, the user will receive data in the high-order 32 bits of each word of the message. If off, the user will receive data in all 36 bits of each word of the message.
B18-35: Special Queue Header

AC2: Address where extended message is to be stored

RETURNS +1: Failure, error code in AC1
+2: Success, message block stored at address given in AC2

The RCVIM JSYS will block until the message is received.

See SNDIM JSYS for a description of the message format.

RCVIM ERROR MNEMONICS:

SQX1: Special network queue handle out of range
SQX2: Special network queue not assigned

TOPS-20 MONITOR CALLS
(RCVIN%)

Receives an Internet datagram. Internet queues are assigned by ASNIQ%.

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: Flags in the left half and an Internet queue handle in the right half.

AC2: Address of message buffer

AC3: Not used, must be 0

RETURNS +1: Failure, with error code in AC1

+2: Success

Flags:

| Bits | Symbol | Meaning |
|------|--------|---------|
|------|--------|---------|

| | | |
|----|--------|--|
| B0 | RIQ%NW | If set, causes RCVIN% to take the error return rather than wait for a message. |
|----|--------|--|

Message Buffer

| Word | Symbol | Meaning |
|------|--------|---------|
|------|--------|---------|

| | | |
|---|--------|---|
| 0 | .INQBH | Maximum length of the message buffer (including this word) in the right half. On return, the monitor fills in the actual length of the message plus one (counting the count word) in the left half. |
|---|--------|---|

| | | |
|---|--------|---|
| 1 | .INQIH | First word of the IP header and message |
|---|--------|---|

RCVIN% ERROR MNEMONICS:

SQX1: Special network queue handle out of range

SQX2: Special network queue not assigned

SNDIX1: Invalid message size

SNDIX2: Insufficient system resources (no buffers available)

SNDIX3: Illegal to specify NCP lines 0 - 72

SNDIX4: Invalid header value for this queue

SNDIX5: IMP down

TOPS-20 MONITOR CALLS
(RCVOK%)

Allows the access-control program (written by the installation) to service an approval request in the GETOK% request queue after a user program has issued a GETOK% JSYS.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Address of argument block

AC2: Length of argument block

RETURNS +1: Always

Argument Block (returned):

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .RCFCJ | Function code,,job number of requestor |
| 1 | .RCUNO | User number |
| 2 | .RCCDR | Connected directory |
| 3 | .RCRQN | Request number |
| 4 | .RCNUA | # args actually passed to RCVOK% block,,# user args supplied in user block |
| 5 | .RCARA | Address of user arguments |
| 6 | .RCCAP | Capabilities enabled |
| 7 | .RCTER | Controlling terminal number (not device designator); or -1 if controlling terminal is detached |
| 10 | .RCRJB | Requested job number |
| | | i-17;11 User arguments |
| . | | .. |
| . | | .. |
| 11+n | | .. |

The argument block returned contains two major segments, the job section, which contains information about the job that issued the GETOK% JSYS, and the user argument section, which contains the arguments the user supplied with the GETOK% call. The user argument section immediately follows the job section. However, as the job section's length may grow with future releases of TOPS-20, the access-control program should extract the address of the user argument section from word .RCARA of the RCVOK% argument block. The following sequence of instructions illustrates how to index through the user argument section of the RCVOK% argument block:

```

;Build AOBJN pointer
HLRZ   T1,ARGBLK+.RCNUA      ;Get # user args passed
MOVN   T1,T1                 ;Negate
HRLZS  T1                    ;Move to left half-word
HRR    T1,ARGBLK+.RCARA     ;Get address of user args

```

TOPS-20 MONITOR CALLS
(RCVOK%)

```
LP:      MOVE      T2,(T1)                ;Get user argument
        ...
        ...
        AOBJN     T1,LP
```

If the access-control program wishes to reject the requested access, the program returns an error code in AC2. It can also provide an error string, which is copied to the caller of GETOK% if the caller has provided a byte pointer for it.

Generates an illegal instruction interrupt on error conditions below.

RCVOK% ERROR MNEMONICS:

```
CAPX1:    WHEEL or OPERATOR capability required
GOKER3:   JSYS not executed within ACJ fork
```

Reads input from the primary input designator (.PRIIN) into the caller's address space. Input is read until either a break character is encountered or the given byte count is exhausted, whichever occurs first. Output generated as a result of character editing is output to the primary output designator (.PRIOU).

The RDTTY call handles the following editing functions:

1. Delete the last character input (DELETE).
2. Delete back to the last punctuation character (CTRL/W).
3. Delete back to the beginning of the current line or, if the current line is empty, back to the beginning of the previous line (CTRL/U).
4. Retype the current line from its beginning or, if the current line is empty, retype the previous line (CTRL/R).
5. Accept the next character without regard to its usual meaning (CTRL/V).

TOPS-20 MONITOR CALLS
(RDTTY)

By handling these functions, the RDTTY call serves as an interface between the terminal and the user program.

ACCEPTS IN AC1: Byte pointer to string in caller's address space where input is to be placed

AC2: B0(RD%BRK) Break on CTRL/Z or ESC.
B1(RD%TOP) Break on CTRL/G, CTRL/L, CTRL/Z, ESC, carriage return, line feed.
B2(RD%PUN) Break on punctuation (see below).
B3(RD%BEL) Break on end of line (carriage return and line feed, or line feed only).
B4(RD%CRF) Suppress a carriage return and return a line feed only.
B5(RD%RND) Return to user program if user tries to delete beyond beginning of the input buffer (for example, user types a CTRL/U or DELETE past the first character in the buffer). If this bit is not set, the call rings the terminal's bell and waits for more input.
B7(RD%RIE) Return to user program if input buffer is empty. If this bit is not set, the call waits for more input.
B9(RD%BEG) Return to the user program if the user attempts to edit beyond the beginning of the input buffer.
B10(RD%RAI) Convert lowercase input to uppercase input.
B11(RD%SUI) Suppress CTRL/U indication (do not print XXX, and on display terminals, do not delete the characters from the screen).
B15(RD%NED) Suppress the editing functions of editing characters (for example, CTRL-R, CTRL-U) that are in the user-supplied break mask.
B18-35 Number of bytes available in the string. The input is terminated when this count is exhausted, even if the specified break character has not yet been typed.

If the left half of AC2 is 0, the input is terminated on end of line only.

AC3: Byte pointer to prompting-text (CTRL/R buffer), or 0 if no text. This text, followed by any text in the input buffer, is output if the user types CTRL/R in his first line of input. If no CTRL/R text exists or the user types CTRL/R on other than the first line of input, only the text on the current line will be output.

TOPS-20 MONITOR CALLS
(RDTTY)

RETURNS +1: Failure, error code in AC1

 +2: Success, updated byte pointer in AC1, appropriate bits set in the left half of AC2, and updated count of available bytes in the right half of AC2

The bits returned in the left half of AC2 on a successful return are:

B12(RD%BTM) Break character terminated the input. If this bit is not set, the input was terminated because the byte count was exhausted.

B13(RD%BFE) Control was returned to the program because the user tried to delete beyond the beginning of the input buffer and RD%RND was on in the call.

B14(RD%BLR) The backup limit for editing was reached.

NOTE

Bits not described are reserved for use by the monitor. The state of these bits on completion of the RDTTY call is undefined.

The punctuation break character set (RD%PUN) is as follows:

| | |
|---------------|---------------------|
| CTRL/A-CTRL/F | ASCII codes 34-36 |
| CTRL/H-CTRL/I | ASCII codes 40-57 |
| CTRL/K | ASCII codes 72-100 |
| CTRL/N-CTRL/Q | ASCII codes 133-140 |
| CTRL/S-CTRL/T | ASCII codes 173-176 |
| CTRL/X-CTRL/Y | |

Upon completion of the call, the terminating character is stored in the string, followed by a NULL (unless the byte count was exhausted). Also, any CTRL/V, along with the character following it, is stored in the string.

RDTTY ERROR MNEMONICS:

RDTX1: Invalid string pointer

IOX11: Quota exceeded

IOX34: Disk full

IOX35: Unable to allocate disk - structure damaged

TOPS-20 MONITOR CALLS
(RELD)

Releases one or all devices assigned to the job. When a device is released by the job, the resource allocator receives an IPCF packet. (See the ALLOC monitor call description for the format of the packet sent to the allocator.)

ACCEPTS IN AC1: Device designator, or -1 to release all devices assigned to this job

RETURNS +1: Failure, error code in AC1

+2: Success

The ASND monitor call can be used to assign a device to the caller.

If this JSYS is issued for a device on which the user has an open JFN, an error will be returned.

RELD ERROR MNEMONICS:

DEVX1: Invalid device designator

DEVX2: Device already assigned to another job

DEVX6: Job has open JFN on device

Releases ownership of an Internet queue so that other jobs can assign it. Internet queues are assigned by ASNIQ%.

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: An Internet queue handle, or -1 for all Internet queue handles, or a job process handle

AC2: Not used, must be 0

AC3: Not used, must be 0

RETURNS +1: Failure, with error code in AC1

+2: Success

TOPS-20 MONITOR CALLS
(RELIQ%)

RELIQ% ERROR MNEMONICS:

SQX1: Special network queue handle out of range
SQX2: Special network queue not assigned

Deassigns the TCP/IP special message queue. (The LGOUT JSYS deassigns all special message queues.) All pending messages relative to the specified queue(s) are discarded. Internet special message queues are assigned by ASNSQ%.

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: Special queue handle (returned by ASNSQ), or -1 to deassign all special queues.

RETURNS +1: Always

RELSQ functions as a no-op if an unassigned queue is specified in AC1.

Closes all files at or below the current process and releases all JFNs; kills all inferior processes; clears the PSI for the current process; sets TT%WKF, TT%WKN, TT%WKP, TT%WKA, TT%ECO and .TTASC of the controlling terminal's JFN mode word; releases all PIDs of the current process; dequeues all ENQ requests for the current process, clears PA1050's entry vector; clears any software traps set with SWTRP%, and, releases all process handles inferior to the current process or killed with KFORK.

RETURNS +1: Always

The RESET monitor call performs the following:

1. Closes all files at or below the current process and releases all JFNs. If a file is nonexistent (has never been closed), it is closed and then expunged.

TOPS-20 MONITOR CALLS
(RESET)

2. Kills all inferior processes.
3. Clears the current process's software interrupt system. The channel table and priority level table addresses remain unchanged from any previous settings.
4. Sets the following fields of the controlling terminal's JFN mode word (see Section 2.4.9.1):

TT%WAK(B18-23) to wake up on every character
TT%ECO(B24) to cause echoing
.TTASI(B29) to translate both echo and output (ASCII data mode)

Remaining fields of the mode word are not changed.

5. Releases all of the current process's PIDs.
6. Dequeues all of the current process's ENQ requests.
7. Clears the compatibility package's entry vector.
8. Releases all process handles that can be released. (See the RFRKH call description.)

Returns the ACs of the specified process.

ACCEPTS IN AC1: Process handle

AC2: Address of the beginning of a 20-word (octal) table in the caller's address space where the AC values of the specified process are to be stored

RETURNS +1: Always

The SFACS monitor call can be used to set the ACs for a specified process.

Generates an illegal instruction interrupt on error conditions below.

RFACS ERROR MNEMONICS:

FRKH1: Invalid process handle

TOPS-20 MONITOR CALLS
(RFACS)

FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX4: Process is running
FRKHX8: Illegal to manipulate an execute-only process

Returns the byte size for a specific opening of a file. (See the OPENF or SFBSZ call description for setting the byte size.)

ACCEPTS IN AC1: JFN

RETURNS +1: Failure, error code in AC1
+2: Success, byte size right-justified in AC2

RFBSZ ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open

Returns the control character output control (CCOC) words for the specified terminal. (See Section 2.4.9.2.)

ACCEPTS IN AC1: File designator

RETURNS +1: Always, with output control words in AC2 and AC3

The CCOC words consist of 2-bit bytes, each byte representing the output control for one of the ASCII codes 0-37. If the given designator is not associated with a terminal, the CCOC words are returned in AC2 and AC3 with each 2-bit byte containing a value of 2 (send actual code and account format action).

TOPS-20 MONITOR CALLS
(RFCOC)

The SFCOC monitor call can be used to set the CCOC words for a specified terminal.

Generates an illegal instruction interrupt on error conditions below.

RFCOC ERROR MNEMONICS:

TTYX01: Line is not active

Returns the JFN mode word associated with the specified file. (See Section 2.4.9.1.) The MTOPR monitor call should be used to return the page length and width fields, especially when the fields have values greater than 127. The RFMOD call returns these fields as 1 when their values are greater than 127.

ACCEPTS IN AC1: Source designator

RETURNS +1: Always, with mode word in AC2

If the designator is not a terminal, the RFMOD call returns in AC2 a word in the following format

7B3+^D66B10+^D72B17+ 4 mode bits from the OPENF for the designator

This setting of the left half of AC2 indicates that the designator has mechanical form feed, mechanical tab, lower case, page length of 66, and page width of 72.

The SFMOD and STPAR monitor calls can be used to set various fields of the JFN mode word.

RFMOD ERROR MNEMONICS:

TTYX01: Line is not active

TOPS-20 MONITOR CALLS
(RFORK)

Resumes one or more processes that had been directly frozen. This monitor call does not resume a process that has been indirectly frozen. (See Section 2.7.3.1.) Also, the RFORK call cannot be used to resume a process that is suspended because of a monitor call intercept. (See the UTFRK call.)

ACCEPTS IN AC1: Process handle

RETURNS +1: Always

The RFORK monitor call is a no-op if the referenced process(s) was not directly frozen.

The FFORK monitor call can be used to freeze one or more processes.

Generates an illegal instruction interrupt on error conditions below.

RFORK ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Returns the current position of the specified terminal's pointer. (See Section 2.4.9.1 for information on page lengths and widths of terminals.)

ACCEPTS IN AC1: Device designator

RETURNS +1: Always, with AC2 containing position within a page (line number) in the left half, and position within a line (column number) in the right half

AC2 contains 0 if the designator is not associated with a terminal.

The SFPOS monitor call can be used to set the position of the terminal's pointer.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(RFPOS)

RFPOS ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Returns the current position of the specified file's pointer.

ACCEPTS IN AC1: JFN

RETURNS +1: Failure, error code in AC1
+2: Success, byte number in AC2

The SFPTR monitor call can be used to set the position of the file's pointer.

RFPTR ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open

Releases the specified handle of a process. A handle can be released only if it describes either an existent process inferior to at least one other process in the job or a process that has been killed via KFORK (a nonexistent process).

ACCEPTS IN AC1: Process handle, or -1 to release all relative handles that can be released

RETURNS +1: Failure, error code in AC1
+2: Success

TOPS-20 MONITOR CALLS
(RFRKH)

The process handles released when AC1 is -1 are the ones released on a RESET or a KFORK monitor call.

RFRKH ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Returns the status of the specified process.

SHORT FORM:

ACCEPTS IN AC1: 0,,process handle

RETURNS +1: Always, with the status word in AC1 and the PC in AC2

Flags:

B0-17 Unused, must be zero.

The process status word has the following format:

B0(RF%FRZ) The process is frozen. If this bit is off, the process is not frozen.

B1-17(RF%STS) The status code for the process. The following values are possible:

| Value | Symbol | Meaning |
|-------|--------|--|
| 0 | .RFRUN | The process is runnable. |
| 1 | .RFIO | The process is dismissed for I/O. |
| 2 | .RFHLT | The process is dismissed by voluntary process termination (HFORK or HALTF) or was never started. |

TOPS-20 MONITOR CALLS
(RFSTS)

| | | |
|----|--------|--|
| 3 | .RFFPT | The process is dismissed by forced process termination. Forced termination occurs when bit 17(SC%FRZ) of the process capability word is not set. |
| 4 | .RFWAT | The process is dismissed waiting for another process to terminate. |
| 5 | .RFSLP | The process is dismissed for a specified amount of time. |
| 6 | .RFTRP | The process is dismissed because it attempted to execute a call on which an intercept has been set by its superior (via the TFORK call). |
| 7 | .RFABK | The process is dismissed because it encountered an instruction on which an address break was set (by means of the ADBRK call). |
| 10 | .RFSIG | The process is dismissed because it attempted to perform I/O on the signal JFN. |

B18-35(RF%SIC) The number of the software interrupt channel that caused the forced process termination.

The RFSTS call returns with -1 (fullword) in AC1 if the specified handle is assigned but refers to a deleted process. The call generates an illegal instruction interrupt if the handle is unassigned.

LONG FORM:

ACCEPTS IN AC1: Flags,,process handle

 AC2: Address of status return block (used for long form only)

RETURNS +1: Always

TOPS-20 MONITOR CALLS
(RFSTS)

Flags:

B0 RF%LNG Long form call (must be on)

B1-17 Unused, must be zero.

In the long form call, RF%LNG is set in AC1 and AC2 contains the address of a status-return block. On the return, AC1 and AC2 are not modified. The status-return block has the following format:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .RFCNT | Count of words returned in this block in the left half, and count of maximum number of words to return in right half (including this word). The right half of this word is specified by the user. |
| 1 | .RFPSW | Process status word. This word has the same format as AC1 on a return from a short call. If a valid, but unassigned, process handle was specified in AC1, then this word contains -1 and no other words are returned. |
| 2 | .RFPFL | Process PC flags. These are the same flags returned in AC2 on a short call. |
| 3 | .RFPPC | Process PC. This is the address; no flags are returned in this word. |
| 4 | .RFSFL | Status flag word. |

Flags:

| Bit | Symbol | Meaning |
|-----|--------|-------------------------|
| B0 | RF%EXO | Process is execute-only |

Generates an illegal instruction interrupt on error conditions below.

RFSTS ERROR MNEMONICS:

DECRSV: DEC-reserved bits not zero
 FRKHX1: Invalid process handle
 FRKHX2: Illegal to manipulate a superior process
 FRKHX3: Invalid use of multiple process handle

TOPS-20 MONITOR CALLS
(RFTAD)

Returns the dates and times associated with the specified file.

ACCEPTS IN AC1: Source designator

AC2: Address of argument block

AC3: Length of argument block

RETURNS +1: Always, with dates returned in the argument block

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .RSWRT | Internal date and time file was last written. |
| 1 | .RSCRV | Internal date and time file was created. |
| 2 | .RSREF | Internal date and time file was last referenced. |
| 3 | .RSCRE | System date and time of last write by the monitor. (The COPY and RENAME commands in the EXEC change this word, for example.) |
| 4 | .RSTDT | Tape-write date and time for archived or migrated files. |
| 5 | .RSNET | Online expiration date and time. May be a date and time (in internal format) or an interval (in days). Intervals are limited to half-word values. |
| 6 | .RSFET | Offline expiration date and time. May be a date and time (in internal format) or an interval (in days). Intervals are limited to half-word values. |

On a successful return, the values for the number of words specified in AC3 are returned in the argument block. Words in the argument block contain -1 if any one of the following occurs:

1. The corresponding date does not exist for the file.
2. The designator is not associated with a file.
3. The corresponding date is not currently assigned (that is, the argument block contains more than 4 words).

The following table illustrates which JSYSS set the file dates and times:

TOPS-20 MONITOR CALLS
(RFTAD)

| Word | GTJFN | OPENF Read | OPENF Write | CLOSF Write | SFTAD | RNAMF | ARCF |
|--------|-------|---------------|----------------|----------------|-------|-------|------|
| .RSWRT | - | - | Set | - | Set | FDB | - |
| .RSCRV | Set | - | - | - | Set | FDB | - |
| .RSREF | - | Set | - | - | Set | Set | - |
| .RSCRE | Set | - | - | Set | Set* | FDB | - |
| .RSTDT | - | - | - | - | Set* | FDB | Set* |
| .RSNET | - | - | - | - | Set | FDB | - |
| .RSFET | - | - | - | - | Set | FDB | - |

LEGEND:

- * Requires WHEEL or OPERATOR capability enabled.
- FDB This word copied from source FDB to destination FDB.

Generates an illegal instruction interrupt on error conditions below.

RFTAD ERROR MNEMONICS:

- DESX1: Invalid source/destination designator
- DESX3: JFN is not assigned
- DESX7: Illegal use of parse-only JFN or output wildcard-designators

Inputs a byte nonsequentially (random byte input) from the specified file. The size of the byte is that given in the OPENF call. The RIN call can be used only when reading data from disk files.

ACCEPTS IN AC1: JFN

AC3: Byte number within the file

RETURNS +1: Always, with the byte right-justified in AC2

If the end of the file is reached, AC2 contains 0. The program can process this end-of-file condition if an ERJMP or ERCAL is the next instruction following the RIN call. Upon successful execution of the call, the file's pointer is updated for subsequent I/O to the file.

The ROUT monitor call can be used to output a byte nonsequentially to a specified file.

TOPS-20 MONITOR CALLS
(RIN)

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

RIN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
IOX1: File is not open for reading
IOX3: Illegal to change pointer for this opening of file
IOX4: End of file reached
IOX5: Device or data error

Returns the channel and priority level table addresses for the specified process. (See Section 2.6.3.) These table addresses are set by the SIR monitor call. The process must run in one section of memory. To obtain the addresses of the channel and priority tables for a process that runs in multiple sections, use the XRIR% monitor call. (See also the XSIR% monitor call.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with the priority level table address in the left half of AC2, and the channel table address in the right half of AC2

AC2 contains 0 if the SIR monitor call has not been executed by the designated process.

Generates an illegal instruction interrupt on error conditions below.

RIR ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

TOPS-20 MONITOR CALLS
(RIRCM)

Returns the mask for reserved software interrupt channels for the specified process. A process is able to read its own or its inferiors' channel masks.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with the reserved channel mask for the specified process in AC2

The SIRCM monitor call can be used to set the mask for reserved software interrupt channels.

Generates an illegal instruction interrupt on error conditions below.

RIRCM ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Releases the specified JFNs. A JFN cannot be released unless it either has never been opened or has already been closed. Also, a JFN cannot be released if it is currently being assigned by a process, unless that process is the same as the one executing the RLJFN and is not at interrupt level. The GS%ASG bit returned from a GTSTS call for the JFN indicates if the JFN is currently being assigned.

ACCEPTS IN AC1: JFN, or -1 to release all JFNs created by this process or its inferiors that do not specify open files

RETURNS +1: Failure, error code in AC1
+2: Success

RLJFN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
RJFN1: File is not closed

TOPS-20 MONITOR CALLS
(RLJFN)

RJFNX2: JFN is being used to accumulate filename
RJFNX3: JFN is not accessible by this process
OPNX1: File is already open

Acquires a handle on a page in a process to determine the access allowed for that page.

ACCEPTS IN AC1: Process handle in the left half, and a page number within the process in the right half

RETURNS +1: Always, with a handle on the page in AC1, and access information in AC2. The handle in AC1 is a process/file designator in the left half and a page number in the right half. This is called a page handle.

The access information returned in AC2 is as follows:

- B2(RM%RD) read access allowed
- B3(RM%WR) write access allowed
- B4(RM%EX) execute access allowed
- B5(RM%PEX) page exists
- B9(RM%CPY) copy-on-write access

If the page supplied in the call does not exist, RMAP returns a -1 in AC1 and a zero in AC2.

Generates an illegal instruction interrupt on error conditions below.

RMAP ERROR MNEMONICS:

FRKHX1: Invalid process handle

Renames an existing file. The JFNs of both the existing file and the new file specification must be closed.

TOPS-20 MONITOR CALLS
(RNAME)

ACCEPTS IN AC1: JFN of existing file to be renamed (source file)

AC2: JFN of new file specification (destination file specification)

RETURNS +1: Failure, error code in AC1

+2: Success, JFN in AC1 is released, and the JFN in AC2 is associated with the file under its new file specification

If the JFN of the new file specification already refers to an existing file, the existing file's contents are expunged.

When a file is renamed, many of the attributes of the existing file are given to the renamed file. The settings of the following words in the FDB (see Section 2.2.8) are copied from the existing file to the renamed file.

| | |
|------|---|
| Word | .FBCTL (FB%LNG, FB%DIR, FB%NOD, FB%BAT, FB%FCF) |
| Word | .FBADR |
| Word | .FBCRE |
| Word | .FBGEN (FB%DRN) |
| Word | .FBBYV (FB%BSZ, FB%MOD, FB%PGC) |
| Word | .FBSIZ |
| Word | .FBCRV |
| Word | .FBWRT |
| Word | .FBREF |
| Word | .FBCNT |
| Word | .FBUSW |

Note that the setting of FB%PRM (permanent file) does not get copied. Thus, if a file with bit FB%PRM on is renamed, the renamed file has FB%PRM off. The existing file is left in a deleted state with its contents empty but its FDB existent.

Renaming a file with tape information (an archived or migrated file) carries the tape information to the new file name. Renames which would effectively destroy a file with archive status will fail.

RNAME ERROR MNEMONICS:

DESX1: Invalid source/destination designator

DESX3: JFN is not assigned

DESX4: Invalid use of terminal designator or string pointer

DESX7: Illegal use of parse-only JFN or output wildcard-designators

OPNX1: File is already open

RNAMEX1: Files are not on same device

RNAMEX2: Destination file expunged

RNAMEX3: Write or owner access to destination file required

RNAMEX4: Quota exceeded in destination of rename

TOPS-20 MONITOR CALLS
(RNAMEF)

RNAMEX5: Destination file is not closed
RNAMEX6: Destination file has bad page table
RNAMEX7: Source file expunged
RNAMEX8: Write or owner access to source file required
RNAMEX9: Source file is nonexistent
RNMX10: Source file is not closed
RNMX11: Source file has bad page table
RNMX12: Illegal to rename to self
RNMX13: Insufficient system resources

Outputs a byte nonsequentially (random byte output) to the specified file. The size of the byte is that given in the OPENF call for the JFN. The ROUT call can be used only when writing data to disk files.

ACCEPTS IN AC1: JFN

AC2: The byte to be output, right-justified

AC3: The byte number within the file

RETURNS +1: Always

Upon successful execution of the call, the file's pointer is updated for subsequent I/O to the file.

The RIN monitor call can be used to input a byte nonsequentially from a specified file.

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

ROUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
IOX2: File is not opened for writing
IOX3: Illegal to change pointer for this opening of file
IOX5: Device or data error
IOX6: Illegal to write beyond absolute end of file

TOPS-20 MONITOR CALLS
(ROUT)

IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Returns the accessibility of a page.

ACCEPTS IN AC1: Process/file designator in the left half, and page number within the process or file in the right half

RETURNS +1: Always, with AC2 containing the following information:

- B2(PA%RD) Read access allowed
- B3(PA%WT) Write access allowed
- B4(PA%EX) Execute access allowed
- B5(PA%PEX) Page exists
- B6(PA%IND) Indirect pointer
- B9(PA%CPY) Copy-on-write
- B10(PA%PRV) Private page
- B20(P1%RD) Read access allowed in first pointer
- B21(P1%WT) Write access allowed in first pointer
- B22(P1%EX) Execute access allowed in first pointer
- B23(P1%PEX) Page exists in first pointer
- B27(P1%CPY) Copy-on-write in first pointer

The bits in the left half are the result of tracing any indirect pointer chains, and the bits in the right half contain information about the first pointer (the one in the map directly indicated by the argument) only.

The left half and right half information will be different only if an indirect pointer was encountered in the first map. In this case, B6(PA%IND) is set, the left access is less than or equal to the right half access; and B9(PA%CPY) is set if it was found set at any level.

The bits B5(PA%PEX) and B10(PA%PRV) always refer to the last pointer (first nonindirect pointer) encountered.

The SPACS monitor call can be used to set the accessibility of a page.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(RPACS)

RPACS ERROR MNEMONICS:

ARGX06: Invalid page number
DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
DESX8: File is not on disk
FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

Returns the capabilities for the specified process. (See Section 2.7.1 for the description of the capability word.)

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with capabilities possible for this process in AC2, and capabilities enabled for this process in AC3

The EPCAP monitor call can be used to enable the capabilities of a process.

Generates an illegal instruction interrupt on error conditions below.

RPCAP ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX3: Invalid use of multiple process handle

Places a text string in, or reads a text string from, the job's rescan buffer (an area of storage in the Job Storage Block). This facility allows a program to receive information that will be used as primary input for another program before this other program reads input from the terminal.

TOPS-20 MONITOR CALLS
(RSCAN)

The RSCAN call has two steps: the acceptance and the use of the text string. Each step has a different calling sequence. The first step is to accept the text string to be used as input and to place this string in the rescan buffer. The calling sequence for this step specifies, in AC1, a pointer to the text string to be input. Note that the string stored in the rescan buffer is terminated by a null byte.

The second step is to make the string available to the program, which can read the string by means of the BIN call. The calling sequence for this second step specifies a function code of 0(.RSINI) in AC1. This code indicates that the last string entered at command level from the terminal is available for reading.

The program executing the RSCAN call can determine when the data has been read by issuing the function code 1(.RSCNT), which returns the number of characters remaining in the buffer.

In other words, the first RSCAN call, specifying a new text string, stores the string in the rescan buffer, but does not cause it to be read. A second RSCAN call must be given before the string can be read.

This second RSCAN causes the system to provide input from the most recent string stored, and can be given only once. After this second RSCAN call, nothing will be read from the rescan buffer until another RSCAN call specifies a different text string. In addition, the job receives input from the rescan buffer only if the source for input in the BIN call is the JFN of the controlling terminal. If the source for input is other than the controlling terminal, input will not come from the rescan buffer.

ACCEPTS IN AC1: Byte pointer to a new text string, or 0 in the left half and function code in the right half

RETURNS +1: Failure, error code in AC1

+2: Success

The defined functions are as follows:

| Function | Symbol | Meaning |
|----------|--------|--|
| 0 | .RSINI | Make the data in the buffer available as input to any process in the current job that is reading data from its controlling terminal. |
| 1 | .RSCNT | Return the number of characters remaining to be read in the buffer. This function does not cause data to be read; it is used to |

TOPS-20 MONITOR CALLS
(RSCAN)

determine when all the data has been read after making the data available.

On a successful return, AC1 contains an updated byte pointer if a pointer was given in the call. Otherwise, AC1 contains either the number of characters in the rescan buffer, or 0 if there are no characters.

To clear the RSCAN buffer, supply a byte pointer (in AC1) to a null string.

RSCAN ERROR MNEMONICS:

RSCNX2: Invalid function code

Reads a section map, and provides information about the mapping of one section of a fork's memory.

ACCEPTS IN AC1: Fork handle,,section number

RETURNS +1: Always, with map information in AC1 and access information in AC2

The map information returned in AC1 can be the following:

- 1 No current mapping present
- 0 The mapping is a private section
- n,,m Where n is a fork handle or a JFN, and m is a section number. If n is a fork handle, the mapping is an indirect or shared map=ping to another fork's sec=tion. If n is a JFN, the mapping is a shared map=ping to a file sec=tion. These are called section handles.

The access in=for=ma=tion bits returned in AC2 are the following:

- B2(SM%RD) Read access is allowed
- B3(SM%WR) Write access is allowed
- B4(SM%EX) Execute access is allowed

TOPS-20 MONITOR CALLS
(RSMAP%)

B5(PA%PEX) The section exists

B6(SM%IND) The section was created using an indirect pointer.

Generates an illegal instruction interrupt on error conditions below.

RSMAP% ERROR MNEMONICS:

ARGX23: Invalid section number

ARGX28: Not available on this system

Returns the handle of the process that was suspended because of a monitor call intercept and the monitor call that the process was attempting to execute. The superior process monitoring the intercepts can receive only one interrupt at a time. Thus, the superior process should execute the RTFRK call after receiving an interrupt to identify the process that caused the interrupt.

The system maintains a queue of the processes that have been suspended and that are waiting to interrupt the superior process monitoring the intercepts. The RTFRK call advances the processes on the queue; and if the call is not executed, subsequent interrupts are not generated.

See the description of the TFORK JSYS for more information on the monitor call intercept facility.

RETURNS +1: Always, with AC1 containing the handle of the process that generated the interrupt, and AC2 containing the monitor call instruction that caused the process to be suspended. If no process is currently suspended because of a monitor call intercept, AC1 and AC2 contain 0 on return.

Because the process handle returned in AC1 is a relative process handle, it is possible that a process is currently suspended, but that all relative handles are in use. In this case, the caller should release a relative process handle with the RFRKH call and then reissue the RTFRK call.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(RTFRK)

RTFRK ERROR MNEMONICS:

FRKH6: All relative process handles in use

Reads the terminal interrupt word (see Section 2.6.6) for the specified process or the entire job, and returns the terminal interrupt word mask.

ACCEPTS IN AC1: B0(RT%DIM) Return the mask for deferred terminal interrupts

B18-35 Process handle, or -5 for entire job
(RT%PRH)

RETURNS +1: always, with the terminal interrupt mask in AC2, and the deferred terminal interrupt mask in AC3. The deferred interrupt mask is returned only if both B0(RT%DIM) is on and the right half of AC1 indicates a specific process.

The STIW monitor call can be used to set the terminal interrupt word masks.

Generates an illegal instruction interrupt on error conditions below.

RTIW ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Returns the run time of the specified process or of the entire job.

ACCEPTS IN AC1: Process handle, or .FHJOB (-5) for the entire job

TOPS-20 MONITOR CALLS
(RUNTM)

RETURNS +1: Always, with runtime (in milliseconds) right-justified in AC1, a divisor to convert time to seconds in AC2, and console time (in milliseconds) in AC3. AC2 always contains 1000; thus, it is not necessary to examine its contents.

Generates an illegal instruction interrupt on error conditions below.

RUNTM ERROR MNEMONICS:

FRKHX1: Invalid process handle
RUNTX1: Invalid process handle -3 or -4

Returns the word mask for the interrupts waiting on software channels for the specified process.

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with

AC1 containing a 36-bit word with bit n on, meaning that an interrupt on channel n is waiting.

AC2 containing the status of the interrupts in progress. Bit n on in the left half means an interrupt of priority level n occurring during execution of user code is in progress. Bit 18+n on in the right half means an interrupt of priority level n occurring during execution of monitor code is in progress.

Generates an illegal instruction interrupt on error conditions below.

RWM ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

TOPS-20 MONITOR CALLS
(RWSET)

Releases the working set by removing all of the current process's pages from its working set. The pages are moved to secondary storage and are not preloaded the next time the process is swapped in. This operation is invisible to the user.

RETURNS +1: Always

Sets the account to which the specified file is to be charged.

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: JFN

 AC2: Account number in bits 3-35 if bits 0-2 contain 5. Otherwise, contains a byte pointer to an account string in the address space of caller. If a null byte is not seen, the string is terminated after 39 characters are processed.

RETURNS +1: Failure, error code in AC1
 +2: Success, updated string pointer in AC2

If the account validation facility is enabled, the SACTF call verifies the account given and returns an error if it is not valid for the caller.

The GACTF monitor call can be used to obtain the account designator to which a file is being charged.

SACTF ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
SACTX1: File is not on multiple-directory device
SACTX2: Insufficient system resources (Job Storage Block full)
SACTX3: Directory requires numeric account
SACTX4: Write or owner access required

TOPS-20 MONITOR CALLS
(SACTF)

VACCX0: Invalid account
VACCX1: Account string exceeds 39 characters
VACCX2: Account has expired

Saves, in nonsharable format, pages of a process in the specified file. The process must run in one section of memory. (See Section 2.8.1 for the format of a nonsharable save file. See the SSAVE monitor call for saving processes in sharable format.) This file can then be copied into a given process with the GET monitor call.

ACCEPTS IN AC1: Process handle in the left half, and JFN in the right half

AC2: One table entry, or 0 in the left half and pointer to the table in the right half (see below)

RETURNS +1: Always

The table has words in the format: length of the area to save in the left half and address of the first word to save in the right half. The table is terminated by a 0 word.

Nonexistent pages are not saved. The SAVE call also does not save the accumulators. Thus, it is possible to save all assigned nonzero memory in section zero or the current section with the table entry 777760,,20 in AC2.

The SAVE call does not save section numbers as parts of addresses, so all addresses are section-relative. Furthermore, the SAVE call saves only the section in which the call is executed.

The SAVE call closes and releases the given JFN.

Can cause several software interrupts or process terminations on certain file conditions.

Generates an illegal instruction interrupt on error conditions below.

SAVE ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle

TOPS-20 MONITOR CALLS
(SAVE)

FRKH8: Illegal to manipulate an execute-only process
SAVX1: Illegal to save files on this device
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

All file errors can also occur.

NOTE

This JSYS is unsupported and is reserved for DIGITAL diagnostics only. The information returned may change in a future release.

WARNING: This JSYS can cause a system crash. Use with extreme caution.

Provides an interface to the System Communications Service (SCS) layer of the System Communications Architecture (SCA), allowing connection management, data transfer, and the exchange of hardware/software configuration information between processes on different systems connected via the CI.

RESTRICTIONS: Requires WHEEL, OPERATOR, MAINTENANCE, or NET WIZARD capability enabled.

ACCEPTS IN AC1: Function code

AC2: Address of argument block

RETURNS +1: Always, with returned data in argument block; generates an illegal instruction trap on failure.

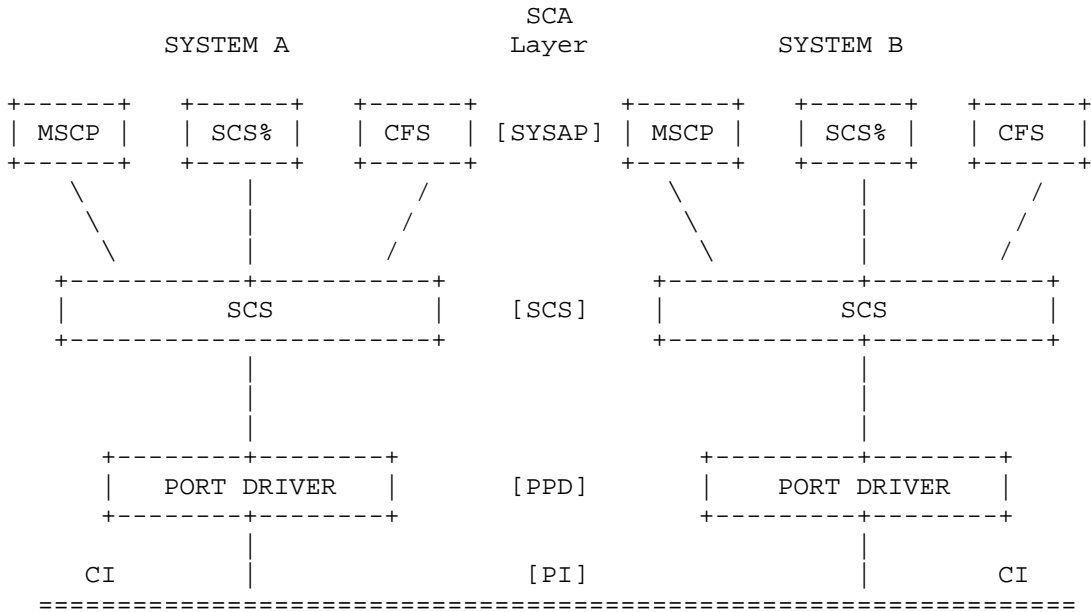
SCA OVERVIEW

SCA is a systems communications architecture, in contrast to a network communications architecture such as DNA. SCS is the systems communications service, a layer of the SCA, which provides communication between processes on different systems connected via the CI (Computer Interconnect).

SCA is a multi-layer protocol, providing a set of connections between hosts on a CI. The layers of SCA are described as follows:

TOPS-20 MONITOR CALLS
(SCS%)

- Layer 3 the System Applications (SYSAP) layer represents the users of SCS, primarily software modules such as CFS (the Common File System) and MSCP (the Mass Storage Control Protocol).
- Layer 2 the Systems Communications Service (SCS) layer provides the process and system addressing, connection management, and flow control necessary to multiplex the basic port/port driver data services among multiple users.
- Layer 1 the Port/Port Driver (PPD) layer controls the Physical Interconnect layer and provides sequential data transfers between ports on the PI.
- Layer 0 the Physical Interconnect (PI) layer supplies a multi-access or point-to-point interconnect, eliminating the need for complex routing facilities in SCA. This is the hardware layer.



SCA Buffers

The same pools of buffers are used for all system applications (SYSAPs). There are two buffer pools: one for datagrams and one for messages. The caller must specify a particular buffer address in the argument blocks of the queue buffer functions. The specified buffer is placed in a pool with all other buffers available to receive incoming data. When the port has a datagram or message to store, it takes the first empty buffer from the appropriate free list, and returns the selected buffer name in the appropriate word of the argument block.

TOPS-20 MONITOR CALLS (SCS%)

Buffers are restricted to one of two sizes: 150 (decimal) words for datagram buffers, and a maximum of 44 (decimal) words for message buffers. Function .SSRBS can be used to return the buffer sizes.

SCA Function Arguments

The following definitions apply to all SCS% function arguments:

ASCII source/destination process strings contain the name of the local (source) process or remote (destination) process. These strings must end on a null byte, and may be no longer than 16 bytes, not including the null byte. Byte size must be at least 7-bit, but may be larger. 7-bit ASCII strings may be defined with the MACRO-20 ASCIZ pseudo-op.

Connection data is left-justified, 32-bit words of data to be sent out with the connection request to the remote (destination) system. The connection data is specified by the user as part of a connect or accept function. Word .SQCDT (.SQCDA) is the address of four contiguous words (SQ%CDT) in the user's address space that are sent to the other side of the connection in the connect or accept. These four words can be used as the user desires. Note that the monitor will copy SQ%CDT words of connection data whether or not the calling program has specified the maximum, so a full block should be allocated.

Messages are data packets with guaranteed delivery. The text for a message is limited to 44 36-bit words. The text must be left justified, word aligned, 8-bit bytes for industry-compatible mode. Datagrams are data packets with no delivery guarantee. They are delivered on a best effort basis. The text for a datagram sent in industry-compatible mode must be packed in left-justified, word aligned, 8-bit bytes, and may be up to 150 words.

The optional path specification (OPS) allows the calling program to send a particular datagram or message over a particular hardware cable (path). The OPS is specified in B30-35(SC%OPS) of word .SQFLG in the function argument block.

The event queue is a record of events about which the calling program wishes to be notified. The caller receives an interrupt when the first event is placed on an empty queue; thereafter, events will be placed on the end of the queue without further notice to the caller. The calling program must empty the queue upon receiving the interrupt.

SCA Interrupts

All notification of SCA events happen on four PSI channels:

TOPS-20 MONITOR CALLS
(SCS%)

1. datagram available
2. message available
3. DMA transfer complete
4. all other SCA events, including virtual circuit closure, connection management events, and all port and SCA-related errors

To enable channels for SCA interrupts, the calling program must execute the .SSAIC function of SCS%, as well as doing all of the normal procedures required to enable the PSI system for TOPS-20. (See Section 2.6.)

DMA

Direct Memory Access (DMA) refers to the ability of a peripheral device to place data into memory or get data from memory without intervention from the processor.

With SCS%, data may be placed directly in memory by mapping a DMA buffer. Each DMA buffer consists of segments which contain a contiguous set of 36-bit words within the calling program's working set. Segments may not cross a page boundary and therefore, may not be more than one page long. Once a buffer has been mapped for a DMA transfer, the contents of that buffer may not be changed until the DMA transfer has been acknowledged complete. If the contents of the buffer are modified prior to the acknowledgement, the modified buffer may be transferred, and the original contents lost.

After the DMA transfer has been acknowledged complete, the calling program may unmap the DMA buffer. Note that unmapping any DMA buffer prior to the acknowledgement can have severe repercussions for the calling program and its environs. The calling process does not have to unmap DMA buffers between data transfers, but must unmap a buffer which will not be used further. Unless unmapped, DMA buffers will remain mapped until the next RESET or CLZFF monitor call or process deletion.

SCS% FUNCTION CODES

| Code | Symbol | Function |
|------|--------|---|
| 0 | .SSCON | Request a connection with another node on the CI. SCS% will return as soon as the connection request has been sent. The calling process will be notified by PSI interrupt when the request is granted, or if the request fails. |

TOPS-20 MONITOR CALLS
(SCS%)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQSPN | Byte pointer to ASCII source process name |
| 2 | .SQDPN | Byte pointer to ASCII destination process name |
| 3 | .SQSYS | B0-17 Node number of destination B18-35 high order 6 bits of connect ID |
| 4 | .SQCDT | Address of connection data |
| 5 | .SQAMC | Address of first buffer on message buffer chain |
| 6 | .SQADC | Address of first buffer on datagram buffer chain |
| 7 | .SQRCI | Returned connect ID |

The length of the argument block is given by symbol .LBCON.

- 1 .SSLIS Listen for a connection; the calling process is notified via PSI interrupt when connection heard.

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQSPN | Byte pointer to ASCII source process name |
| 2 | .SQDPN | Byte pointer to ASCII destination process name; to listen for any process on a particular system, set the destination process to -1. See word .SQSYS. |
| 3 | .SQSYS | B0-17 Node number of destination B18-35 high order 6 bits of connect ID To listen for a particular process (specified in .SQDPN) on any system, set the destination node number to -1. If both .SQDPN and the left half of .SQSYS are set to -1, then any connect request not destined for a particular process will match the listen. |
| 4 | .SQLCI | Returned connect ID |

The length of the argument block is given by symbol .LBLIS.

- 2 .SSREJ Reject a connection with another node on the CI

TOPS-20 MONITOR CALLS
(SCS%)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQREJ | Rejection code indicating the reason for rejecting the connection |

The length of the argument block is given by symbol
.LBREJ.

3 .SSDIS Disconnect and close a connection

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQDIS | Disconnect code indicating the reason for closing the connection |

The length of the argument block is given by symbol
.LBDIS.

4 .SSSDG Send a datagram

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQAPT | Address of datagram text |
| 3 | .SQLPT | Length of datagram text in words for high density and in bytes for industry compatible |
| 4 | .SQFLG | <flags>B29!<OPS>B35 B1(SC%MOD) Mode flag: high density if set industry compatible if clear B30-35(SC%OPS) Optional path specification 0 = .SSAPS field auto path select 1 = .SSPTA use path A 2 = .SSPTB use path B .SSLOW Lowest value for SC%OPS field .SSHGH Highest value for SC%OPS field |

TOPS-20 MONITOR CALLS
(SCS%)

The length of the argument block is given by symbol .LBSDG.

- 5 .SSQRD Queue buffer(s) to receive a datagram; the first word of each buffer is the address of the next buffer; the first word of the last buffer contains 0 as the address of the next buffer

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQAFB | Address of first buffer in buffer chain |

The length of the argument block is given by symbol .LBQRD.

- 6 .SSSMG Send a message to a remote node

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQAPT | Address of message text |
| 3 | .SQLPT | Length of message (in 8-bit bytes for industry compatible mode and in words for high density mode) |
| 4 | .SQFLG | <flags>B29!<OPS>B35 B1(SC%MOD) Mode flag: high density if set industry compatible if clear |

The length of the argument block is given by symbol .LBSMG.

- 7 .SSQRM Queue buffer(s) to receive a message; the first word of each buffer is the address of the next buffer; the first word of the last buffer contains 0 as the address of the next buffer. Buffer size is fixed at 38 36-bit words.

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQAFB | Address of first message buffer in message buffer chain |

TOPS-20 MONITOR CALLS
(SCS%)

The length of the argument block is given by symbol .LBQRM.

- 10 .SSCSP Return information about the state of a connection
- | | | |
|---|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQCST | Connection state (returned) |
| 3 | .SQDCI | Destination connect ID (returned) |
| 4 | .SQBDN | Byte pointer indicating location to start destination process name; may be either "real" byte pointer, or "generic" byte pointer (-1,,STRING); if a generic byte pointer is used, the string will be written as 16 word-aligned 8-bit bytes. (updated byte pointer returned) |
| 5 | .SQSBI | Node number (returned) |
| 6 | .SQREA | <source disconnect code>,,<destination disconnect code> (returned) |

The length of the argument block is given by symbol .LBCSP.

- 11 .SSRCD Return configuration data about remote system
- | Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID (optional); if zero, contents of word .SQOSB are used to determine the target system (see below) |
| 2 | .SQOSB | Node number (optional); either .SQCID or .SQOSB must be specified, but only one of the two may be specified |
| 3 | .SQVCS | <virtual circuit state>,,<port number> (returned) Virtual circuit states 0 = VC.CLO closed 1 = VC.STS start sent 2 = VC.STR start receive 3 = VC.OPN open |
| 4-5 | .SQSAD | Remote system address (8, 8-bit bytes returned) |
| 6 | .SQMDD | Maximum datagram size at destination (returned) |
| 7 | .SQMDM | Maximum message size at destination (returned) |

TOPS-20 MONITOR CALLS
(SCS%)

| | | |
|-------|--------|---|
| 10 | .SQDST | Software type at destination (4 bytes, 8-bit ASCII string returned) |
| 11 | .SQDSV | Software version at destination (4 bytes, 8-bit ASCII string returned) |
| 12-13 | .SQDSE | Software edit level at destination (8 bytes, 8-bit ASCII string returned) |
| 14 | .SQDHT | Hardware type code at destination (4 bytes, 8-bit ASCII string returned) |
| 15-17 | .SQDHV | Hardware version at destination (12 bytes, 8-bit ASCII string returned) |
| 20-21 | .SQNNM | Destination port node name (8 bytes, 8-bit ASCII string returned) |
| 22 | .SQPCW | Port characteristics word (returned) |
| 23 | .SQLPN | Local port number (RH20 channel number of CI-20) (returned) |

The length of the argument block is given by symbol .LBRCD.

12 .SSSTS Return status information about a connection

| Word | Symbol | Contents |
|------|------------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQFST | <flags>,,<connect state code> |
| | | Flags: |
| | B0(SC%MSA) | message available - there is at least one message available for this connection. |
| | B1(SC%DGA) | datagram available - there is at least one datagram available for this connection. |
| | B2(SC%DTA) | DMA transfer complete - at least one DMA transfer has completed. |
| | B3(SC%EVA) | event available - at least one event is pending. |
| | | Connect state codes: |
| | 1(SQ%CLO) | closed |
| | 2(SQ%LIS) | listening for connection |
| | 3(SQ%CSE) | connect request sent |
| | 4(SQ%CRE) | connect request received |
| | 5(SQ%CAK) | connect acknowledge received |

TOPS-20 MONITOR CALLS
(SCS%)

6(SQ%ACS) accept request sent
 7(SQ%RJS) reject request sent
 10(SQ%OPN) connection open
 11(SQ%DSE) disconnect request sent
 12(SQ%DRE) disconnect request received
 13(SQ%DAK) disconnect response received
 14(SQ%DMC) waiting for disconnect response
 SQ%HIS highest value for a connect state

3 .SQSBR <reserved>,,<node number of remote>

The length of the argument block is given by symbol .LBSTS.

13 .SSRMG Receive a message; returns message text for either the calling fork or the specified connection

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID or -1; if this word contains -1, then the message returned is the first one found for the calling fork; if this word contains any other value (that is, a connect ID), then the message returned is the first one found for the specified connection. In either case, if no message is found, an illegal instruction trap is generated. |
| 2 | .SQARB | Address of returned message buffer (returned); this address is an address in the caller's working set that was previously specified with function .SSQRM, and in which the monitor has placed the returned message. If no .SSQRM has been executed, an illegal instruction trap is generated. |
| 3 | .SQDFL | B0-17(SC%FRM) Flags B18-35(SC%NRM) Node number of remote system B1(SC%MOD) Mode flag: high density if set industry compatible if clear |
| 4 | .SQLRP | Length of returned message; this length is returned in bytes for an |

TOPS-20 MONITOR CALLS
(SCS%)

industry-compatible message, and in words for a high density mode message. (See word .SQDFL above.)

The length of the argument block is given by symbol .LBRMG.

14 .SSMAP Associate a block of memory with an DMA buffer name to be used in DMA data transfers

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQXFL | Flags and Mode field 32(SQ%CVD) Do not clear the valid bit 33(SQ%WRT) Read/Write if set host memory is writable 34-35(SQ%DMD) Mode field 0 = SQ%DIC industry compatible mode 1 = SQ%DCD core dump 2 = SQ%DHD high density mode 3 = SQ%ILL disallowed value |
| 2 | .SQBNA | Name of DMA buffer (returned) Followed by buffer length and address pairs |
| | .SQBLN | Length of memory block in bytes for high density and 8-bit bytes for industry compatible (see .SQBAD below). |
| | .SQBAD | Address of memory in calling program's working set for DMA transfer; words .SQBLN and .SQBAD are specified in pairs for each segment of a DMA buffer to be mapped. |

15 .SSUMP Unmap a memory block assigned for DMA transfers

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQNAM | Buffer name (returned by .SSMAP) |

The length of the argument block is given by symbol .LBUMP.

16 .SSSND Transfer data to a remote host

TOPS-20 MONITOR CALLS
(SCS%)

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID for which transfer is to be done |
| 2 | .SQSNM | Buffer name of send buffer |
| 3 | .SQRNM | Buffer name of receive buffer |
| 4 | .SQOFS | <transmit offset>,,<receive offset> The offsets are in words for high density and in bytes for industry compatible. |

The length of the argument block is given by symbol .LBSND.

17 .SSREQ Request delivery of data for specified buffer

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID for which transfer is to be done |
| 2 | .SQSNM | Buffer name of send buffer |
| 3 | .SQRNM | Buffer name of receive buffer |
| 4 | .SQOFS | <transmit offset>,,<receive offset> The offsets are in words for high density and in bytes for industry compatible. |

The length of the argument block is given by symbol .LBREQ.

20 .SSAIC Add interrupt channels for SCA events

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1-4 | | Up to 4 channel descriptor words of the format: <interrupt type code>,,<channel for this code> |

Interrupt type codes:

| | | |
|---|--------|------------------------------------|
| 0 | .SIDGA | interrupt on datagram available |
| 1 | .SIMSA | interrupt on message available |
| 2 | .SIDMA | interrupt on DMA transfer complete |
| 3 | .SIPAN | interrupt on all other events |

TOPS-20 MONITOR CALLS
(SCS%)

A -1 for the channel removes the interrupt type.

22 .SSRDG Receive a datagram; returns datagram text for either the calling fork or the specified connection.

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID or -1; if this word contains -1, the datagram returned is the first one found for the calling fork; if this word contains any other value (that is, a connect ID), the datagram returned is the first one found for the specified connection. |
| 2 | .SQARB | Address of returned datagram buffer (returned); this address is an address in the caller's working set that was previously specified with function .SSQRD, and in which the monitor has placed the returned datagram. If no datagram is found, the content of this word is zero. If no .SSQRD has been executed or if the address is not writable, an illegal instruction trap is generated. |
| 3 | .SQDFL | B0-17(SC%FRM) Flags B18-35(SC%NRM) Node number of remote |
| 4 | .SQLRP | Length of returned datagram; this length is returned in bytes for an industry-compatible datagram, and in words for a high density mode datagram. (See word .SQDFL above.) |

The length of the argument block is given by symbol .LBRDG.

23 .SSACC Accept a connection with another node on the CI that has requested a connection.

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQCDA | Address of 4-word (SQ%CDT) connection data block |

TOPS-20 MONITOR CALLS
(SCS%)

The length of the argument block is given by symbol .LBACC.

- 24 .SSGDE Return the first entry from the data request complete queue and repeat until queue is empty.

Word Symbol Contents

| | | |
|---|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID or -1 |
| 2 | .SQBID | Buffer ID of buffer that completed DMA transfer (returned) |

The length of the argument block is given by symbol .LBGDE.

- 25 .SSEVT Retrieve first entry from event queue; this function must be repeated until the event queue is empty.

Word Symbol Contents

| | | |
|---|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID or -1; if -1, the next event for the calling fork is returned; if connect ID, the next event for the specified connection is returned. CID is returned. |
| 2 | .SQESB | Left half is reserved for DIGITAL. Right half is node number of remote node. |
| 3 | .SQEVT | Event code (see .SQDTA below) |
| 4 | .SQDTA | Event data |

Event codes and data:

| | | |
|---|--------|---|
| 1 | .SEVCC | Virtual circuit broken .SQDTA contains the pertinent node number |
| 2 | .SECTL | Connect to listener .SQDTA contains 4 words (SQ%CDT) of connection data from the remote node |
| 3 | .SECRA | Connection was accepted .SQDTA contains 4 words (SQ%CDT) of connection data from the remote node |
| 4 | .SECRR | Connection was rejected .SQDTA contains the rejection reason code |

TOPS-20 MONITOR CALLS
(SCS%)

| | | |
|----|--------|--|
| 5 | .SEMSC | Message or datagram send complete .SQDTA contains address of sent buffer |
| 6 | .SELCL | Little credit left .SQDTA contains the number of credits required to restore the calling program's credit threshold |
| 7 | .SENWO | Node went offline .SQDTA contains node number of system that went offline |
| 10 | .SENCO | Node came online .SQDTA contains node number of system that came online |
| 11 | .SEOSD | OK to send data .SQDTA is not used |
| 12 | .SERID | Remote initiated disconnect .SQDTA is not used |
| 13 | .SEPBC | Port broke connection .SQDTA is not used |
| 14 | .SECIA | Credit is available .SQDTA is not used |
| 15 | .SEMDC | Maintenance data transfer complete .SQDTA is the buffer name for the transfer |
| | .SEMAX | Maximum event code. |

The length of the argument block is given by symbol .LBEVT.

26 .SSCRD Cancel datagram receive; removes the buffer queued for datagram reception

Word Symbol Contents

| | | |
|---|--------|---|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQADB | Address of buffer to dequeue; must be address of previously queued datagram buffer; if address not found by monitor, causes an illegal instruction trap |

The length of the argument block is given by symbol .LBCRD.

27 .SSCRM Cancel message receive; removes buffer queued for message reception

TOPS-20 MONITOR CALLS
(SCS%)

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | 0,,<block length>; on return <# of words processed>,,<block length> |
| 1 | .SQCID | Connect ID |
| 2 | .SQADB | Address of buffer to dequeue; must be address of previously queued message buffer; if the address is not found by the monitor, illegal instruction trap is generated |

The length of the argument block is given by symbol
.LBCRM.

30 .SSGLN Get local node number

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | <# of words processed>,, <block length> |
| 1 | .SQLNN | local node number |

The length of the argument block is given by symbol
.LBGLN.

35 .SSRBS Return minimum buffer sizes

| Word | Symbol | Contents |
|------|--------|---|
| 0 | .SQLEN | <# of words processed>,, <block length> |
| 1 | .SQLMG | Length in words of smallest allowed message buffer |
| 2 | .SQLDG | Length in words of smallest datagram buffer |

The length of the argument block is given by symbol
.LBRBS.

36 .SSRPS Return path status

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SQLEN | <# of words processed>,,<block length> |
| 1 | .SQRPN | Target node number |
| 2 | .SQRPS | Path status |

B0-17 Path A status
B18-35 Path B status

Status Definition

1 = SC%PGD path is good 0 = SC%PBD path
is bad

TOPS-20 MONITOR CALLS
(SCS%)

The length of the argument block is given by symbol
.LBRPS.

SCS% ERROR MNEMONICS:

SCSBFC: Function code out of range
SCSBTS: Argument block too short
SCSIAB: Invalid argument block address
SCSNSN: No source process name specified on connection request
SCSNBP: Not enough privileges enabled
SCSNSC: No such connect ID
SCSIID: Invalid connect ID
SCSNBA: Internal resources exhausted (No more SCA buffers)
SCSSCP: DMA segment crosses a page boundry
SCSQIE: Queue is empty
SCSFRK: Fork does not own this SCS% data
SCSNMQ: No buffers queued for message reception
SCSISB: Invalid node number
SCSIBP: Invalid byte pointer
SCSNDQ: No datagram buffers queued
SCSENB: Excessive number of buffers in queue request
SCSSTL: DMA buffer segment too long
SCSTMS: Too many DMA buffer segments
SCSNBS: No such buffer
SCSNKP: No known KLIPA on this system
SCSIPC: PSI channel out of range
SCSIPS: Invalid path spec
SCSIST: Invalid SCS% interrupt type
SCSIDM: Invalid DMA transmission mode
SCSIBN: Invalid buffer name
SCSTBF: No slots left in CID tables
SCSBFC: Function code out of range
SCSAAB: Error accessing argument block
SCSDCB: Datagram text crosses a page boundry
SCSNRT: No room in table for address entry
SCSNPA: No packet address
SCSZLP: Zero length packet text
SCSNSD: No such DMA buffer name
SCSDDL: DMA buffer too long
SCSUPC: Unknown PSI code
SCSNBS: Not enough room for SCS headers
SCSIAB: Invalid address in arguments
SCSJBD: No user address found for sent packet
SCSCWS: Connection in incorrect state for function
SCSNEC: Not enough credit
SCSBAS: Internal error, bad argument to subroutine
SCSNEB: Insufficient buffers to fill request
SCSIFL: Invalid forward link in buffer chain

TOPS-20 MONITOR CALLS
(SCTTY)

Redefines the controlling terminal for the specified process and all of its inferiors. The controlling terminal can be redefined at any level in the job's process structure; inferior processes below this level uses this terminal by default as their controlling terminal. Therefore, the controlling terminal of a process is defined to be:

1. The one that has been explicitly defined for it by a SCTTY call.
2. If no terminal has been explicitly defined for the process, the terminal that has been explicitly defined for its closest superior by a SCTTY call.
3. If no SCTTY call has been executed for a superior process, the job's controlling terminal.

The effect of terminal interrupts on a process is dictated by the controlling terminal for the process. This means that processes that have enabled specific terminal characters receives an interrupt when those characters are typed on the controlling terminal. If no SCTTY call has been executed for any process in the job, the controlling terminal for all processes within the job is the job's controlling terminal. (The job's controlling terminal is usually the one used to log in and control the job.) In addition to being the source of all terminal interrupts, the job's controlling terminal serves as the primary I/O designators (see Section 1.2.6) for all processes in the job, unless these designators have been changed for a process.

When a SCTTY call is executed for a process within a job, the controlling terminal and the source of terminal interrupts are changed for that process and all of its inferiors. This group of processes receives interrupts only from the new controlling terminal and no longer from the job's controlling terminal. These processes cannot receive or change terminal interrupts from any other controlling terminals. However, primary I/O continues to be received from and sent to the job's controlling terminal if the primary I/O designators have not been changed. For most applications, the primary I/O designators should be changed with the SPJFN call to correspond to the new controlling terminal.

ACCEPTS IN AC1: Function code in the left half, and process handle in the right half

AC2: Terminal designator

RETURNS +1: Always

The available functions are as follows:

TOPS-20 MONITOR CALLS
(SCTTY)

| Code | Symbol | Meaning |
|------|---------|--|
| 0 | .SECRET | Return the designator of the given process's controlling terminal. The designator is returned in AC2. |
| 1 | .SCSET | Change the given process's controlling terminal to the terminal designated in AC2. The terminal designator cannot refer to the job's controlling terminal. This function also changes the controlling terminal of all processes inferior to the given process. |
| 2 | .SCRST | Reset the given process's controlling terminal to the job's controlling terminal. This function also resets the controlling terminal of all processes inferior to the given process. |

Functions .SCSET and .SCRST require the process to have the SC%SCT capability (see Section 2.7.1) enabled in its capability word.

The SCTTY monitor call cannot be used to change the controlling terminal for the current process or for any process superior to the current process.

Generates an illegal instruction interrupt on error conditions below.

SCTTY ERROR MNEMONICS:

SCTX1: Invalid function code
SCTX2: Terminal already in use as controlling terminal
SCTX3: Illegal to redefine the job's controlling terminal
SCTX4: SC%SCT capability required
FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
DESX1: Invalid source/destination designator
DEVX2: Device already assigned to another job

Sets the entry vector and the UUO locations for the compatibility package.

ACCEPTS IN AC1: Process handle

TOPS-20 MONITOR CALLS
(SCVEC)

AC2: Entry vector length in the left half, and entry vector address in the right half

AC3: UUO location in the left half, and PC location in the right half

RETURNS +1: Always

The compatibility package's entry vector is as follows:

| Word | Symbol | Meaning |
|---------|--------|---|
| 0 | .SVEAD | Entry address for interpreting UUOs |
| 1 | .SVINE | Initial entry for setup and first UUO |
| 2 | .SVGET | Entry for GET share file routine (obsolete) |
| 3 | .SV40 | Address to receive contents of location 40 on the UUO call |
| 4 | .SVRPC | Address to receive the return PC word on the UUO call |
| 5 | .SVMAK | Entry for MAKE share file routine (obsolete) |
| 6 and 7 | .SVCST | Communication for handling CTRL/C, START sequences between the compatibility package and the TOPS-20 Command Language |

The monitor transfers to the address specified in the right half of AC2 on any monitor call whose operation code is 040-077 (a monitor UUO). This transfer occurs after the monitor stores the contents of location 40 and the return PC in the locations specified by the left half and right half of AC3, respectively. The entry vector is retained but is not used by the monitor.

If AC2 is 0, the next UUO causes the compatibility package to be merged into the caller's address space. In this case, the UUO and PC locations are set from words 3 and 4, respectively, of the compatibility package's entry vector.

If AC2 is -1, UUO simulation is disabled, and an occurrence of a UUO is considered an illegal instruction. This action is useful when the user is removing UUOs from a program.

The GCVEC monitor call can be used to obtain the entry vector for the compatibility package.

TOPS-20 MONITOR CALLS
(SCVEC)

SCVEC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate superior process
FRKHX3: Invalid use of multiple process handle
FRKHX4: Process is running
FRKHX8: Illegal to manipulate an execute-only process

Sets the status of a device. (See Section 2.4 for the descriptions of the status bits.) This call requires that the device be opened.

ACCEPTS IN AC1: JFN

AC2: New status bits

RETURNS +1: Always

The SDSTS call is a no-op for devices that do not have device-dependent status bits.

The GDSTS monitor call can be used to obtain the status bits for a particular device.

Generates an illegal instruction interrupt on error conditions below.

SDSTS ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
DESX9: Invalid operation for this device

Sets the entry vector for the Record Management System (RMS). (See the RMS Manual for more information on the Record Management System.)

TOPS-20 MONITOR CALLS
(SDVEC)

RESTRICTIONS: Requires RMS software.

ACCEPTS IN AC1: Process handle

AC2: Entry vector length in the left half, and entry vector address in the right half

RETURNS +1: Always

The Record Management System's entry vector is as follows:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .SDEAD | Entry address for the RMS calls |
| 1 | .SDINE | Initial entry for the first RMS call |
| 2 | .SDVER | Pointer to RMS version block |
| 3 | .SDDMS | Address in which to store the RMS call |
| 4 | .SDRPC | Address in which to store return PC word |

The GDVEC monitor call can be used to obtain the entry vector for RMS.

The XSSEV% monitor call can be used to set an extended special entry vector for RMS entry vectors in nonzero sections.

Generates an illegal instruction interrupt on error conditions below.

SDVEC ERROR MNEMONICS:

ILINS5: RMS facility is not available

FRKH8: Illegal to manipulate an execute-only process

Sets the most recent error condition encountered by a process. This error condition is stored in the process's Process Storage Block.

ACCEPTS IN AC1: Process handle

AC2: Error code that is to be set

RETURNS +1: Always

The GETER monitor call can be used to obtain the most recent error condition encountered by a process.

TOPS-20 MONITOR CALLS
(SETER)

Generates an illegal instruction interrupt on error conditions below.

SETER ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Process is running
FRKHX8: Illegal to manipulate an execute-only process

Sets job parameters for the specified job.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Job number, or -1 for the current job

AC2: Function code

AC3: Value for function

RETURNS +1: Always

The available functions, along with the legal values for these functions, are described below.

| Function | Values | Meaning |
|-----------|-----------|--|
| .SJDEN(0) | | Set default for magnetic tape density. |
| | .SJDDN(0) | System default density |
| | .SJDN2(1) | 200 bits/inch (8.1 rows/mm) |
| | .SJDN5(2) | 556 bits/inch (22.5 rows/mm) |
| | .SJDN8(3) | 800 bits/inch (32.2 rows/mm) |
| | .SJD16(4) | 1600 bits/inch (65.3 rows/mm) |
| | .SJD62(5) | 6250 bits/inch (246 rows/mm) |
| .SJPAR(1) | | Set default for magnetic tape parity. |
| | .SJPRO(0) | Odd parity |
| | .SJPRE(1) | Even parity |
| .SJDM(2) | | Set default for magnetic tape data mode. |

TOPS-20 MONITOR CALLS
(SETJB)

.SJDDM(0) System default data mode
 .SJDMC(1) Dump mode
 .SJDM6(2) SIXBIT byte mode (7-track drives)
 .SJDMA(3) ANSI ASCII mode (7 bits in 8-bit bytes)
 .SJDM8(4) Industry-compatible mode
 .SJDMH(5) High-density mode for TU70 and TU72 tape drives only (nine 8-bit bytes in two words)

.SJRS(3) Set default for magnetic tape record size in bytes. The maximum allowable number of bytes depends on the hardware data mode specified for the drive:

| Data Mode | Maximum Number Bytes |
|---------------------|----------------------|
| default | - |
| dump | 8192 |
| SIXBIT | 49152 |
| ANSI ASCII | 40960 |
| industry compatible | 32768 |
| high density | 8192 |

Note that the SETJB JSYS does not return an error message if the above values are exceeded. However, the OPENF or the first data transfer (whichever is performed first after function .SJDM) fails. Note that MTOPR function .MOSRS can be used to override the default record size specified with SETJB function .SJDM.

.SJDFS(4) Set spooling mode.

.SJSPI(0) Immediate mode spooling
 .SJSPD(1) Deferred mode spooling

.SJSRM(5) Set remark for current job session. AC3 contains a pointer to the session remark, which is updated on a successful return. The first 39 characters of the session remark are placed in the job's Job Storage Block.

.SJT20(6) Indicate if job is at EXEC level or program level.

-1 job is at EXEC level
 0 job is at program level

TOPS-20 MONITOR CALLS
(SETJB)

| | |
|----------------|--|
| .SJDFR(7) | Set job default retrieval. Allows a user to override the system default for OPENF. |
| .SJRFA(0) | Any OPENF of a disk file should fail if file's contents are not on line. This is the system default. |
| .SJRWA(1) | Any OPENF of a disk file should wait for the ARCF JSYS to restore the contents of a file to disk. |
| .SJBAT(10) | Set batch flags and batch stream number |
| OB%WTO(3B1) | Write to operator capabilities |
| .OBALL(0) | WTO (write to operator) and WTOR (write to operator with reply) allowed |
| .OBNWR(1) | No WTR allowed |
| .OBNOM(2) | No message allowed |
| OB%BSS(1B10) | OB%BSN (see below) contains a batch stream number |
| OB%BSN(177B17) | Batch stream number |
| .SJLLO(11) | Set job logical location (node name) |

The SETJB monitor call requires the process to have WHEEL or OPERATOR capability enabled to set parameters for a job other than the current job.

The GETJI monitor call can be used to obtain the job parameters for a specified job.

Generates an illegal instruction interrupt on error conditions below.

SETJB ERROR MNEMONICS:

| | |
|--------|---------------------------------------|
| SJBX1: | Invalid function |
| SJBX2: | Invalid magnetic tape density |
| SJBX3: | Invalid magnetic tape data mode |
| SJBX4: | Invalid job number |
| SJBX5: | Job is not logged in |
| SJBX6: | WHEEL or OPERATOR capability required |
| SJBX7: | Remark exceeds 39 characters |
| SJBX8: | Illegal to perform this function |

TOPS-20 MONITOR CALLS
(SETNM)

Sets the private name of the program being used by the current job. This name is the one printed on SYSTAT listings.

ACCEPTS IN AC1: SIXBIT name used to identify program

RETURNS +1: Always

The GETNM monitor call can be used to obtain the name of the program currently being used.

Sets either the system name or the private name of the program being used by the current job.

ACCEPTS IN AC1: SIXBIT name to be used as the system name. This name is the one used for system statistics.

AC2: SIXBIT name to be used as the private name. This name is the same as the one set with the SETNM call.

RETURNS +1: Failure. (Currently, there are no failure returns defined.)

+2: Success

System program usage statistics are accumulated in the system tables SNames, STimes, and SPflts. (See Section 2.3.2.) To make this possible, the SETSN call must be executed by each job whenever the system program name is changed. In the usual case, the TOPS-20 Command Language handles this. The argument to SETSN should be: for system programs (programs from SYS:), the filename, truncated to six characters and converted to SIXBIT; for private programs, "(PRIV)".

Sets the entry vector of the specified process. The process must run in only one section of memory.

TOPS-20 MONITOR CALLS
(SEVEC)

ACCEPTS IN AC1: Process handle

AC2: Entry vector word (length in the left half and address of first word in the right half), or 0

RETURNS +1: Always

A zero in AC2 removes the entry vector for the process.

The GEVEC monitor call can be used to obtain the process's entry vector.

The XSVEC% monitor call sets the entry vector of a process that runs in a section other than section zero.

Generates an illegal instruction interrupt on error conditions below.

SEVEC ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process
SEVEX1: Entry vector length is not less than 1000

Sets the ACs of the specified process.

ACCEPTS IN AC1: Process handle

AC2: Address of the beginning of a 20(octal) word table in the caller's address space. This table contains the values to be placed into the ACs of the specified process.

RETURNS +1: Always

The specified process must not be running.

The RFACS call can be used to obtain the ACs for a specified process.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(SFACS)

SFACS ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX4: Process is running
FRKHX8: Illegal to manipulate an execute-only process

Resets the byte size for a specific opening of a file. (See the OPENF and RFBSZ calls descriptions.)

ACCEPTS IN AC1: JFN

AC2: Byte size, right-justified

RETURNS +1: Failure, error code in AC1

+2: Success

The SFBSZ monitor call recomputes the EOF limit and the file's pointer based on the new byte size given.

SFBSZ ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX5: File is not open
DESX8: File is not on disk
SFBSX1: Illegal to change byte size for this opening of file
SFBX2: Invalid byte size

Sets the control character output control (CCOC) for the specified terminal, which must be assigned to the caller. (See Section 2.4.9.2 and the RFCOC call description.)

TOPS-20 MONITOR CALLS
(SFCOC)

ACCEPTS IN AC1: File designator

AC2: Control character output control word

AC3: Control character output control word

RETURNS +1: Always

The CCOC words consist of 2-bit bytes, each byte representing the output control for one of the ASCII codes 0-37.

The SFCOC call is a no-op if the designator is not associated with a terminal assigned to the caller.

The RFCOC monitor call can be used to obtain the CCOC words for a specified terminal.

Generates an illegal instruction interrupt on error conditions below.

SFCOC ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Sets the program-related modes for the specified terminal. The modes that can be set by this call are in the following bits of the JFN mode word. (See Section 2.4.9.1.)

B0(TT%OSP) Output suppression control
B18-B23(TT%WAK) Wakeup control
B24(TT%ECO) Echoes on
B28-B29(TT%DAM) Data mode

ACCEPTS IN AC1: File designator

AC2: JFN mode word

RETURNS +1: Always

TOPS-20 MONITOR CALLS
(SFMOD)

The SFMOD call is a no-op if the designator is not associated with a terminal.

The STPAR monitor call can be used to set device-related modes of the JFN mode word, and the RFMOD monitor call can be used to obtain the JFN mode word.

SFMOD ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Starts the specified process in a single section. If the process is frozen, the SFORK call changes the PC but does not resume the process. The RFORK call must be used to resume the process.

ACCEPTS IN AC1: Flags,,process handle

Flags:

SF%CON(1B0) Used to continue a process that has previously halted. If SF%CON is set, the address in AC2 is ignored, and the process continues from where it was halted.

AC2: PC of the process being started. The PC contains flags in the left half and the process starting address in the right half. This call obtains the section number of the PC from the entry vector of the process.

RETURNS +1: Always

The SFRKV monitor call can be used to start a process at a given position in its entry vector.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(SFORK)

SFORK ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX5: Process has not been started
FRKHX8: Illegal to manipulate an execute-only process

Sets the position of the specified terminal's pointer. (See Section 2.4.9.4 for information on page lengths and widths of terminals.)

ACCEPTS IN AC1: File designator

AC2: Position within a page (line number) in the left half, and position with a line (column number) in the right half

RETURNS +1: Always

The SFPOS monitor call is a no-op if the designator is not associated with a terminal or is in any way illegal.

The RFPOS monitor call can be used to obtain the current position of the terminal's pointer.

SFPOS ERROR MNEMONICS:

TTYX01: Line is not active

Sets the position of the specified file's pointer for subsequent I/O to the file. The SFPTR call specifying a certain byte number, followed by a BIN call, has the same effect as a RIN call specifying the same byte number.

ACCEPTS IN AC1: JFN

TOPS-20 MONITOR CALLS
(SFPTR)

AC2: Byte number to which the pointer is to be set, or -1 to set the pointer to the current end of the file.

SF%LSN(1B0) LSN flag bit. If SF%LSN is set, include the LSN as text in the position setting. If SF%LSN is not set, ignore the LSN.

RETURNS +1: Failure, error code in AC1

+2: Success

The following comments concern line sequence numbers (LSNs):

By default, the monitor ignores all LSNs and nulls when doing input from a file. (Nulls are used to insure that the LSN starts on a word boundary.) When the first byte of the file is read, the monitor checks the word containing that byte to see if it is part of an LSN. If it is not, the monitor sets an internal flag that is equivalent to setting OF%PLN in the OPENF. This flag specifies that all bytes will be passed to the user program. If the monitor's internal flag is not set, then LSNs and nulls are suppressed.

If the monitor has not checked the first word of the file (as is the case when a process executes an SFPTR JSYS to move the file byte pointer to a byte in some other word of the file) and the process did not set OF%PLN in the OPENF, then the monitor assumes that the file contains LSNs. LSNs and nulls are not passed to the user program. Thus nulls will be suppressed even if the file contains no LSNs. In this case, if it is desired that nulls should be passed to the user program, then OF%PLN should be set in the OPENF, regardless of whether the file actually contains LSNs.

The RFPTR monitor call can be used to obtain the current position of the file's pointer.

SFPTR ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX8: File is not on disk
SFPTX1: File is not open
SFPTX2: Illegal to reset pointer for this file
SFPTX3: Invalid byte number

TOPS-20 MONITOR CALLS
(SFRKV)

Starts the specified process using the given position in its entry vector.

ACCEPTS IN AC1: Process handle

AC2: Word (0-n) in the entry vector that contains the address to use for the start address. Word 0 is always the primary start address, and word 1 is the reenter address.

RETURNS +1: Always

The process starts execution at the address that is the starting address of the entry vector plus the offset specified in AC2. That location must contain an executable instruction.

If the process has a TOPS-10 format entry vector (JRST in the left half), then the left half of AC2 in the SFRKV call is the start address offset. The only legal offsets are 0 and 1, and they are only legal for entry vector position 0 (start address). Thus, for TOPS-10 entry vectors, the left half of AC2 will be added to the contents of the right half of .JBSA to determine the start address. Entry vector position 0 means "use the contents of the right half of .JBSA (120) as the start address," and position 1 means "use the contents of the right half of .JBREN (124) as the reenter address."

NOTE

It is illegal to use an entry vector position other than 0 or 1 for an execute-only process.

Generates an illegal instruction interrupt on error conditions below.

SFRKV ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX4: Process is running
FRKHX8: Illegal to manipulate an execute-only process
SFRVX1: Invalid position in entry vector

TOPS-20 MONITOR CALLS
(SFTAD)

Sets the dates and times associated with the specified file.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Source designator

AC2: Address of argument block

AC3: Length of argument block

RETURNS +1: Always

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .RSWRT | Internal date and time file was last written. |
| 1 | .RSCRV | Internal date and time file was created. |
| 2 | .RSREF | Internal date and time file was last referenced. |
| 3 | .RSCRE | System date and time of last write by the monitor. (The COPY and RENAME commands in the EXEC change this word, for example.) Requires WHEEL or OPERATOR capability enabled. |
| 4 | .RSTDT | Tape-write date and time of archived or migrated files. Requires WHEEL or OPERATOR capability enabled. |
| 5 | .RSNET | On-line expiration date and time, which can be a date and time (in internal format) or an interval (in days). Intervals are limited to half-word values. Dates, times, and intervals can not exceed system or directory maximums. |
| 6 | .RSFET | Offline expiration date and time, which can be a date and time (in internal format) or an interval (in days). Intervals are limited to half-word values. Dates, times, and intervals can not exceed system or directory maximums. |

For words .RSWRT, .RSCRV, and .RSREF, the new values are checked against the current date and time. Values greater than the current date and time can be set only if the process has WHEEL or OPERATOR capability enabled.

TOPS-20 MONITOR CALLS
(SFTAD)

If the designator represents a device for which dates are meaningless (dates for terminals, for example), or if any value given is -1, the given value is ignored, and the current date, if pertinent, is not changed. If the argument block has more than four words, given values for these words are checked to be in valid format and then ignored, if valid.

The following table illustrates which monitor calls set the file dates and times:

| Word | GTJFN | OPENF Read | OPENF Write | CLOSF Write | SFTAD | RNAMF | ARCF |
|--------|-------|---------------|----------------|----------------|-------|-------|------|
| .RSWRT | - | - | Set | - | Set | FDB | - |
| .RSCRV | Set | - | - | - | Set | FDB | - |
| .RSREF | - | Set | - | - | Set | Set | - |
| .RSCRE | Set | - | - | Set | Set* | FDB | - |
| .RSTDT | - | - | - | - | Set* | FDB | Set* |
| .RSNET | - | - | - | - | Set | FDB | - |
| .RSFET | - | - | - | - | Set | FDB | - |

LEGEND:

- * Requires WHEEL or OPERATOR capability enabled.
- FDB This word copied from source FDB to destination FDB.

The various SFTAD words map to words in the FDB block. (The mnemonic changes from .RS%% to .FB%%.)

The RFTAD monitor call can be used to obtain the dates and times associated with a specified file.

Generates an illegal instruction interrupt on error conditions below.

SFTAD ERROR MNEMONICS:

- ARGX32: On line expiration cannot exceed system or directory maximum
- DESX1: Invalid source/destination designator
- DESX3: JFN is not assigned
- DESX7: Illegal use of parse-only JFN or output wildcard-designators
- DATE6: System date and time not set
- STADX2: Invalid date or time
- CFDBX2: Illegal to change specified bits
- OPNX25: Device is write locked
- CAPX1: WHEEL or OPERATOR capability required

TOPS-20 MONITOR CALLS
(SFUST)

Sets the name of either the author of the file or the user who last wrote to the file.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code in the left half, and JFN of the file in the right half

AC2: Byte pointer to ASCIZ string containing the name

RETURNS +1: Always, with an updated byte pointer in AC2

The defined functions are as follows:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .SFAUT | Set the name of the author of the file. |
| 1 | .SFLWR | Set the name of the user who last wrote the file. |

The GFUST monitor call can be used to return the name of either the author of the file or the user who last wrote the file.

The process must have WHEEL or OPERATOR capability enabled to set the writer's name or to have write or owner access to the file to set the author's name.

Generates an illegal instruction interrupt on error conditions below.

SFUST ERROR MNEMONICS:

SFUSX1: Invalid function
SFUSX2: Insufficient system resources
SFUSX4: File expunged
SFUSX5: Write or owner access required
SFUSX6: No such user name
DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DESX8: File is not on disk
DESX10: Structure is dismounted
CAPX1: WHEEL or OPERATOR capability required

TOPS-20 MONITOR CALLS
(SIBE)

Tests to see if the designated file input buffer is empty.

ACCEPTS IN AC1: Source designator

RETURNS +1: (one of the following is true:)

1. The device is an active terminal and the input buffer is not empty. AC2 contains a count of the bytes remaining in the input buffer.
2. The device is not a terminal, is open for read, and the input buffer is not empty. AC2 contains a count of the bytes remaining in the input buffer.

+2: (one of the following is true:)

1. The device is a non-active terminal. AC2 contains the error code.
2. The device is an active terminal and the input buffer is empty. AC2 contains zero.
3. The device is not a terminal and is not open for read. AC2 contains zero.
4. The device is not a terminal, is open for read, and the input buffer is empty. AC2 contains zero.

The SOBE monitor call can be used to determine if the output buffer is empty, and the SOBF monitor call can be used to determine if the output buffer is full.

SIBE ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

TOPS-20 MONITOR CALLS
(SIN)

Reads a string from the specified source into the caller's address space. The string can be a specified number of bytes, or can be terminated with a specific byte.

ACCEPTS IN AC1: Source designator

AC2: Byte pointer to string in the caller's address space

AC3: Count of number of bytes in string, or 0

AC4: Byte (right-justified) on which to terminate input (optional)

RETURNS +1: Always, with updated byte pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 controls the number of bytes to read.

AC3=0 The string being read is terminated with a 0 byte.

AC3>0 A string of the specified number of bytes is to be read or a string terminated with the byte given in AC4 is to be read, whichever occurs first.

AC3<0 A string of minus the specified number of bytes is to be read.

The contents of AC4 are ignored unless AC3 contains a positive number.

The input is terminated when the byte count becomes 0, the specified terminating byte is reached, the end of the file is reached, or an error occurs during the transfer. The program can process an end-of-file condition if an ERJMP or ERCAL is the next instruction following the SIN call.

After execution of the call, the file's pointer is updated for subsequent I/O to the file. AC2 is updated to point to the last byte read or, if AC3 contained 0, the last nonzero byte read. The count in AC3 is updated toward zero by subtracting the number of bytes read from the number of bytes requested to be read. If the input was terminated by an end-of-file interrupt, AC1 through AC3 are updated (where pertinent) to reflect the number of bytes transferred before the end of the file was reached.

When the SIN call is used to read data from a magnetic tape, the size of the records to read is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) words. The record size must be at least as large as the largest record being read from the tape.

TOPS-20 MONITOR CALLS
(SIN)

The SIN call reads across record boundaries on the tape until it reads the number of bytes specified in AC3. The call gives the data to the program with no indication of tape marks. Thus, if the record is 1000 bytes and a SIN call is given requesting 2000 bytes, it returns two full records to the program.

When reading in reverse, both the number of bytes requested in AC3 and the record size should equal the size of the record on the tape. (See Section 2.4.7 for more information about magnetic tape I/O.)

This call can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

SIN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX1: File is not open for reading
IOX4: End of file reached
IOX5: Device or data error
IOX7: Insufficient system resources (Job Storage Block full)
IOX8: Monitor internal error

Reads a record from the specified device into the caller's address space. The maximum size of the record to read is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) bytes.

ACCEPTS IN AC1: Source designator
AC2: Byte pointer to string in the caller's address space
AC3: Count of number of bytes in string, or 0
AC4: Byte (right-justified) on which to terminate input (optional)
RETURNS +1: Always, with updated byte pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

TOPS-20 MONITOR CALLS
(SINR)

The contents of AC3 and AC4 are interpreted in the same manner as they are in the SIN monitor call.

Each SINR call returns one record to the caller. Thus, the caller can read variable-length records by indicating in AC3 the number of bytes to read. Upon execution of the call, AC3 is updated to reflect the number of bytes read (the number of bytes in the record).

The number of bytes read depends on the number of bytes requested and the record size. When using SINR, the program must set the record size to a value greater than or equal to the actual size of the largest record being read from the tape, or an error (IOX5) will be returned. If the SINR call requests the same number of bytes as the record size, the requested number is given to the caller. When the record size equals the size of the actual record, all bytes in the record are read, and AC3 contains 0 on return. When the record size is larger than the actual record, all bytes of the record are read, but AC3 contains the difference of the number requested and the number read. If the SINR call requests fewer bytes than in the actual record, the requested number is given to the caller, the remaining bytes are discarded, and an error (IOX10) is returned. In all cases, the next request for input begins reading at the first byte of the next record on the tape because a SINR call never reads across record boundaries.

When reading in reverse, the number of bytes requested (that is, the count in AC3) should be at least as large as the size of the record on the tape. If the requested number is smaller, the remaining bytes in the record are discarded from the beginning of the record.

The action taken on a SINR call differs from the action taken on a SIN call. The SIN call reads across record boundaries to read all the bytes in a file. The SINR call does not read across record boundaries and will discard some bytes in the file if the requested number is smaller than the actual record.

For a TCP/IP transmission, SINR will return when a TCP message with the PUSH flag is received, or the byte count is exhausted.

For a DECnet transmission, SINR will read a record and discard any part that does not fit in the user buffer.

Generates an illegal instruction interrupt on error conditions below.

SINR ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open

TOPS-20 MONITOR CALLS
(SINR)

IOX1: File is not open for reading
IOX4: End of file reached
IOX5: Device or data error
IOX7: Insufficient system resources (Job Storage Block full)
IOX8: Monitor internal error
IOX10: Record is longer than user requested

Sets the addresses of the channel and priority level tables for the specified process. (See Section 2.6.3.) The process must run in one section of memory, or Section 0. The tables must also be in that section. To set the table addresses for a process that runs in multiple sections, use the XSIR% monitor call. (See also the XRIR% monitor call.)

ACCEPTS IN AC1: Process handle

AC2: Address of the priority level table in the left half,
and address of the channel table in the right half

RETURNS +1: Always. The addresses in AC2 are stored in the
Process Storage Block.

If the contents of the tables are changed after execution of the SIR call, the new contents will be used on the next interrupt.

The RIR monitor call can be used to obtain the table addresses for a process that runs in a single section.

Generates an illegal instruction interrupt on error conditions below.

SIR ERROR MNEMONICS:

SIRX1: Table address is not greater than 20
FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle
FRKH8: Illegal to manipulate an execute-only process

TOPS-20 MONITOR CALLS
(SIRCM)

Sets the mask for reserved software interrupt channels for the specified inferior process. Conditions occurring on software channels that have the corresponding mask bit set do not generate an interrupt to the inferior process. Instead, the conditions cause the process to terminate or freeze.

ACCEPTS IN AC1: Inferior process handle

AC2: Channel mask with bits set for reserved channels

AC3: Deferred terminal interrupt word

RETURNS +1: Always

The RIRCM monitor call can be used to obtain the mask for reserved software interrupt channels. Although a process can read its own channel mask, it cannot set its own; the SIRCM call can be given only for inferior processes. This call provides a facility for a superior process to monitor an inferior one (for example, illegal instructions, memory traps). However, if the inferior process contains an ERJMP or ERCAL symbol after instructions that generate an interrupt on failure, the ERJMP or ERCAL will prevent the generation of the interrupt. Thus, the superior will not be able to monitor the inferior with the SIRCM call.

Generates an illegal instruction interrupt on error conditions below.

SIRCM ERROR MNEMONICS:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

FRKHX8: Illegal to manipulate an execute-only process

Returns the length of an existing file.

ACCEPTS IN AC1: JFN

RETURNS +1: Failure, error code in AC1

+2: Success, byte count that referenced the last byte written into the file in AC2, and number of pages

TOPS-20 MONITOR CALLS
(SIZEF)

(512 words) in file in AC3. The byte count returned depends on the byte size recorded in the FDB and not on the byte size specified in the OPENF call.

For a file with holes, the byte count in AC2 does not reflect the file's actual size.

The GTFDB monitor call can be used to obtain the byte size in which the file was written.

SIZEF ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
STRX10: Structure is offline

Sets the scheduler priority control word. This word controls the priority of a job and the permissible range of queues that the job may run in. The priority word is set for the top process and for all existing inferior processes. Also, the priority word is passed down to any forks that are created subsequent to the SJPRI call.

RESTRICTIONS: This JSYS is reserved for DIGITAL. Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Job number

AC2: Priority word

RETURNS +1: Always

The priority word has the following format:

B0-17(JP%RTG) is the percentage of CPU resources to be guaranteed for the job. This value may be in the range $0 \leq n \leq 99$.

B18(JP%SYS) is the flag (JP%SYS) that designates the job as a system job. System jobs get a higher priority than all user jobs, and the scheduler gives them all the time they need for execution.

TOPS-20 MONITOR CALLS
(SJPRI)

B24-29(JP%MNQ) is the highest priority queue in which the job can run.

B30-35(JP%MXQ) is the lowest priority queue in which the job can run. This queue is always specified as the desired queue + 1. For example, queue 2 is specified as 3.

Note that the high queue is high in priority but low in numerical value while the low queue is low in priority but high in numerical value.

A priority word containing zero in the left half means no CPU percentage is being requested. A priority word containing zero in the right half means no queue assignments are being requested.

Because this call assigns priority to a job, it is indeterminate how processes within a job that compete for the job's run time will be scheduled. Use of this call for a job containing more than one process implies that the processes must cooperate.

The SPRIW monitor call can be used to set the priority word for a specified process.

Generates an illegal instruction interrupt on error conditions below.

SJPRI ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

SJPRX1: Job is not logged in

Reads or modifies the monitor's scheduler data base.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code

AC2: Address of argument block

RETURNS +1: Always

TOPS-20 MONITOR CALLS
(SKED%)

The available functions are:

| Code | Symbol | Function | | | | | | | | | |
|------|--------|--|------|--------|----------|---|--------|--|---|--------|---------------------------|
| 1 | .SKRBC | <p>Read bias control knob setting. Return a value indicating the setting of the bias control knob. This setting determines whether the scheduler favors compute-bound jobs or interactive jobs.</p> <p>Argument block:</p> <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: left;">.SACNT</td> <td>Count of words in argument block (Including this word)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: left;">.SAKNB</td> <td>Bias control knob setting</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .SACNT | Count of words in argument block (Including this word) | 1 | .SAKNB | Bias control knob setting |
| Word | Symbol | Contents | | | | | | | | | |
| 0 | .SACNT | Count of words in argument block (Including this word) | | | | | | | | | |
| 1 | .SAKNB | Bias control knob setting | | | | | | | | | |
| 2 | .SKSBC | <p>Set bias control setting to the specified value. The setting of this value controls the bias between interactive and compute-bound jobs. The lower the setting, the more interactive jobs are favored. The higher the setting, the more compute-bound jobs are favored. Currently, the value may be an integer n such that $1 \leq n \leq 20$. Requires WHEEL or OPERATOR capability enabled.</p> <p>Argument block:</p> <table border="0" style="margin-left: 2em;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: left;">.SACNT</td> <td>Count of words in argument block (Including this word)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: left;">.SAKNB</td> <td>Bias control knob setting</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .SACNT | Count of words in argument block (Including this word) | 1 | .SAKNB | Bias control knob setting |
| Word | Symbol | Contents | | | | | | | | | |
| 0 | .SACNT | Count of words in argument block (Including this word) | | | | | | | | | |
| 1 | .SAKNB | Bias control knob setting | | | | | | | | | |
| 3 | .SKRCS | <p>Read class parameters. Returns the following values:</p> <ol style="list-style-type: none"> 1. Class of the job 2. Share of the processor allocated for this class. The share is returned as a floating-point value n, such that $0 \leq n \leq 1$. 3. Amount of processor actually used by the class. The amount used is returned as a floating-point value n, such that $0 \leq n \leq 1$. 4. 1 minute load average. The load average = (J/P) where J is the number of CPU-runnable jobs in | | | | | | | | | |

TOPS-20 MONITOR CALLS
(SKED%)

the class for the time period and P is the fraction of CPU allocated to the class. Thus 3 jobs running in a 50% class would produce a load average of 6.

- 5. 5 minute load average
- 6. 15 minute load average

Argument block:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SACNT | Count of words in argument block (Including this word) |
| 1 | .SACLS | Class |
| 2 | .SASHR | Share |
| 3 | .SAUSE | Use |
| 4 | .SA1ML | 1 minute load average |
| 5 | .SA5ML | 5 minute load average |
| 6 | .SA15L | 15 minute load average |

- 4 .SKSCS Set class parameters (as described above). Requires WHEEL or OPERATOR capability.

Argument block:

| Word | Symbol | Contents |
|------|--------|--|
| 0 | .SACNT | Count of words in argument block (Including this word) |
| 1 | .SACLS | Class |
| 2 | .SASHR | Share |
| 3 | .SAWA | Windfall allocation |

- 5 .SKICS Start or stop the class scheduler. If the class scheduler is being started, this function also specifies the mode in which class-to-user assignments are made and whether windfall is to be allocated to the active classes or withheld from the active classes. Requires WHEEL or OPERATOR capability.

TOPS-20 MONITOR CALLS
(SKED%)

| Word | Symbol | Contents | | | | | | | | | | | | |
|------|--------|---|------|--------|----------|----|--------|---|----|--------|-------------------------------|----|--------|---------------------|
| 0 | .SACNT | Count of words in argument block (Including this word) | | | | | | | | | | | | |
| 1 | .SACTL | Control flags | | | | | | | | | | | | |
| | | The flags are as follows: | | | | | | | | | | | | |
| | | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>B0</td> <td>SK%ACT</td> <td>Class by accounts</td> </tr> <tr> <td>B1</td> <td>SK%WDF</td> <td>Withhold windfall</td> </tr> <tr> <td>B2</td> <td>SK%STP</td> <td>Class scheduler off</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | B0 | SK%ACT | Class by accounts | B1 | SK%WDF | Withhold windfall | B2 | SK%STP | Class scheduler off |
| Bit | Symbol | Meaning | | | | | | | | | | | | |
| B0 | SK%ACT | Class by accounts | | | | | | | | | | | | |
| B1 | SK%WDF | Withhold windfall | | | | | | | | | | | | |
| B2 | SK%STP | Class scheduler off | | | | | | | | | | | | |
| 6 | .SKSCJ | Set the class of a job. This function takes a pair of numbers, the job to set and the desired class. If setting the class of the calling job, this function is not privileged. If setting the class of another job, it requires WHEEL or OPERATOR capability enabled. In either case, the job must be allowed to reside in the selected class. The calling job may be designated by -1. | | | | | | | | | | | | |
| | | Argument block: | | | | | | | | | | | | |
| | | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>.SACNT</td> <td>Count of words in argument block (Including this word)</td> </tr> <tr> <td>1</td> <td>.SAJOB</td> <td>Job number</td> </tr> <tr> <td>2</td> <td>.SAJCL</td> <td>Class of job</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .SACNT | Count of words in argument block (Including this word) | 1 | .SAJOB | Job number | 2 | .SAJCL | Class of job |
| Word | Symbol | Contents | | | | | | | | | | | | |
| 0 | .SACNT | Count of words in argument block (Including this word) | | | | | | | | | | | | |
| 1 | .SAJOB | Job number | | | | | | | | | | | | |
| 2 | .SAJCL | Class of job | | | | | | | | | | | | |
| 7 | .SKRJP | Read class parameters for a job | | | | | | | | | | | | |
| | | Argument block: | | | | | | | | | | | | |
| | | <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;">Word</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Contents</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>.SACNT</td> <td>Count of words in argument block (including this word)</td> </tr> <tr> <td>1</td> <td>.SAJOB</td> <td>Job number (provided by user)</td> </tr> </tbody> </table> | Word | Symbol | Contents | 0 | .SACNT | Count of words in argument block (including this word) | 1 | .SAJOB | Job number (provided by user) | | | |
| Word | Symbol | Contents | | | | | | | | | | | | |
| 0 | .SACNT | Count of words in argument block (including this word) | | | | | | | | | | | | |
| 1 | .SAJOB | Job number (provided by user) | | | | | | | | | | | | |

TOPS-20 MONITOR CALLS
(SKED%)

| | | | |
|----|--------|--|---|
| | 2 | .SAJCL | Returns class of job |
| | 3 | .SAJSH | Returns job share |
| | 4 | .SAJUS | Returns job utilization |
| | 5 | .SACSH | Returns class share |
| | 6 | .SACLU | Returns class utilization |
| 10 | .SKBCR | Read the class setting for batch jobs. A -1 indicates that there is no special class for batch jobs. | |
| | | Argument block: | |
| | | Word | Symbol Contents |
| | | 0 | .SACNT Count of words in argument block (Including this word) |
| | | 1 | .SABCL Batch class |
| 11 | .SKBCS | Set batch class. Specifies the class in which all batch jobs will run. A -1 indicates no special class for batch jobs. If this value is specified, it overrides the valid classes for any user running a batch job. Requires WHEEL or OPERATOR capability. | |
| | | Argument block: | |
| | | Word | Symbol Contents |
| | | 0 | .SACNT Count of words in argument block (Including this word) |
| | | 1 | .SABCL Batch class |
| 12 | .SKBBG | Run all batch jobs in the "dregs" queue. The dregs queue is a special queue whose processes are only allowed to run when no normally scheduled processes are available to run. Requires WHEEL or OPERATOR capability. | |
| | | This function applies only if the class scheduler is not being used. The argument is either 0 (clear) or nonzero (set). A nonzero indicates that batch jobs should be run in the "dregs" queue. | |
| | | Argument block: | |

TOPS-20 MONITOR CALLS
(SKED%)

| Word | Symbol | Contents | | | | | | | | | | | | | | | |
|------|--------|---|-----|--------|---------|----|--------|-------------------|----|--------|-------------------|----|--------|---------------------|----|--------|---|
| 0 | .SACNT | Count of words in argument block (Including this word) | | | | | | | | | | | | | | | |
| 1 | .SADRG | Flag word 0 = don't run in dregs queue nonzero = run in dregs queue | | | | | | | | | | | | | | | |
| 14 | .SKRCV | Read status | | | | | | | | | | | | | | | |
| | | Argument block: | | | | | | | | | | | | | | | |
| Word | Symbol | Contents | | | | | | | | | | | | | | | |
| 0 | .SACNT | Count of words in argument block (Including this word) | | | | | | | | | | | | | | | |
| 1 | .SACTL | Flags The flags are as follows: | | | | | | | | | | | | | | | |
| | | <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left;">Bit</th> <th style="text-align: left;">Symbol</th> <th style="text-align: left;">Meaning</th> </tr> </thead> <tbody> <tr> <td>B0</td> <td>SK%ACT</td> <td>Class by accounts</td> </tr> <tr> <td>B1</td> <td>SK%WDF</td> <td>Withhold windfall</td> </tr> <tr> <td>B2</td> <td>SK%STP</td> <td>Class scheduler off</td> </tr> <tr> <td>B3</td> <td>SK%DRG</td> <td>Batch jobs are being run in dregs queue</td> </tr> </tbody> </table> | Bit | Symbol | Meaning | B0 | SK%ACT | Class by accounts | B1 | SK%WDF | Withhold windfall | B2 | SK%STP | Class scheduler off | B3 | SK%DRG | Batch jobs are being run in dregs queue |
| Bit | Symbol | Meaning | | | | | | | | | | | | | | | |
| B0 | SK%ACT | Class by accounts | | | | | | | | | | | | | | | |
| B1 | SK%WDF | Withhold windfall | | | | | | | | | | | | | | | |
| B2 | SK%STP | Class scheduler off | | | | | | | | | | | | | | | |
| B3 | SK%DRG | Batch jobs are being run in dregs queue | | | | | | | | | | | | | | | |

SKED% ERROR MNEMONICS:

ARGX02: Invalid function
 ARGX04: Argument block too small
 ARGX08: No such job
 ARGX15: Job is not logged in
 ARGX25: Invalid class
 ARGX29: Invalid class share
 ARGX30: Invalid KNOB value
 ARGX31: Class scheduler already enabled
 CAPX1: WHEEL or OPERATOR capability required
 SKDX1: Cannot change class

TOPS-20 MONITOR CALLS
(SKPIR)

Tests to see if the software interrupt system is enabled for the specified process.

ACCEPTS IN AC1: Process handle

RETURNS +1: Failure, software interrupt system is off

+2: Success, software interrupt system is on

The EIR monitor call is used to enable the software interrupt system, and the DIR monitor call is used to disable the system.

Generates an illegal instruction interrupt on error conditions below.

SKPIR ERROR MNEMONICS:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

Maps one or more contiguous sections of memory. This call removes any existing mapping from the section or sections named as the destination. To learn the contents of a section map, use the RSMAP% monitor call. The four SMAP% functions are discussed below.

Case I: Mapping File Sections to a Process

This function maps one or more sections of a file to a process. All pages that exist in the source sections are mapped to the destination sections.

To map a process section to a file, use the PMAP monitor call.

ACCEPTS IN AC1: Source identifier: JFN,,file section number

AC2: Destination identifier: fork handle,,process section number

AC3: Flags,,count

TOPS-20 MONITOR CALLS
(SMAP%)

The flags determine access to the destination section, and the count is the number of contiguous sections to be mapped. The count must be between 1 and 37 (octal). The flags are as follows.

| | |
|-----------|--|
| B2(SM%RD) | Allow read access |
| B3(SM%WR) | Allow write access |
| B4(SM%EX) | Allow execute access |
| B18-35 | The number of sections to map. This number must be between 1 and 37. |

RETURNS +1: Always

Case II: Mapping Process Sections to a Process

This function maps one or more sections of memory from one process to another. All pages that exist in the source sections are mapped to the destination sections.

ACCEPTS IN AC1: Source identifier: fork handle,,section number

AC2: Destination identifier: fork handle,,section number

AC3: Flags,,count

The flags determine access to the destination section, and the count is the number of contiguous sections to be mapped. This count must be between 1 and 37. All source sections that exist are mapped to destination sections. The flags are as follows.

| | |
|------------|--|
| B2(SM%RD) | Allow read access |
| B3(SM%WR) | Allow write access |
| B4(SM%EX) | Allow execute access |
| B6(SM%IND) | Map the destination section using an indirect section pointer. Once the destination section map is created, the indirect section pointer causes the destination section map to change in exactly the same way that the source section map changes. |

TOPS-20 MONITOR CALLS
(SMAP%)

B18-35 Count of the number of contiguous
 sections to be mapped.

RETURNS +1: Always

If you map a source section into a destination section with SM%IND set, SMAP% creates the destination section using an indirect pointer. This means that the destination section will contain all pages that exist in the source section, and the contents of the destination pages will be identical to the contents of the source pages.

In addition, changes that occur in the source section map after SMAP% creates the destination section cause the same changes to be made in the destination section map. This ensures that both the source section and the destination section contain the same data.

If SM%IND is not set, SMAP% creates the new section using a shared pointer. After SMAP% maps the destination section, changes that occur in the source section's map do not cause any change in the destination section's map. Thus after a short time the source and destination sections might contain different data.

If you request a shared pointer (SM%IND not set) to the destination section, what happens depends on the contents of the source section when the SMAP% call executes. The outcome is one of the following.

1. If the source section does not exist, the SMAP% call fails.
2. If the source is a private section, a mapping to the private section is established, and the destination process is co-owner of the private section.
3. If the source section contains a file section, the source section is mapped to the destination section. Although files do not actually have section boundaries, this monitor call views them as having sections that consist of 512 contiguous pages. Each file section starts with a page number that is an integer multiple of 512.
4. If the source section map is made by means of an indirect section pointer, SMAP% follows that pointer until the source section is found to be nonexistent, a private section, or a section of a file.

Case III: Creating a Section

This function creates a new, private section. It does not map any pages into the new section.

A process must use SMAP% to create a nonzero section before

TOPS-20 MONITOR CALLS
(SMAP%)

referencing such a section. A reference to a nonexistent section fails with an illegal memory reference error. Note, however, that if a process uses PMAP to map a page to a nonexistent section, the monitor creates a private section and the PMAP succeeds.

ACCEPTS IN AC1: 0

AC2: Destination identifier: fork handle,,section number

AC3: Flags,,count

The flags determine access to the destination section, and the count is the number of contiguous private sections to be created. This count must be between 1 and 37. The flags are as follows.

B2(SM%RD) Allow read access

B3(SM%WR) Allow write access to the created section. This function sets this bit by default to avoid the creation of a read-only or execute-only private section.

B4(SM%EX) Allow execute access to the created section.

B6(SM%IND) Create the section using an indirect pointer.

B18-35 Count of the number of contiguous sections to be created. This number must be between 1 and 37.

RETURNS +1: Always

Case IV: Deleting Process Sections

This function removes (unmaps) a section or several contiguous sections of a process.

ACCEPTS IN AC1: -1

AC2: Destination identifier: fork handle,,section number

AC3: 0,,count

The count is the number of contiguous sections to be unmapped. This number must be between 1 and 37.

RETURNS +1: Always

If the section being removed (unmapped) was created with a shared pointer, and if the removing fork is not the owner of the section, then SMAP% decrements the share count for the section and deletes the

TOPS-20 MONITOR CALLS
(SMAP%)

shared pointer. This is always true when the memory sections being deleted contain file sections.

If the pointer being deleted is the last pointer to a private section, then SMAP% clears the page table for that section. But if the owning fork attempts to unmap a private section to which other forks have shared or indirect pointers, the SMAP% call fails.

Generates an illegal instruction interrupt on error conditions below.

SMAP% ERROR MNEMONICS:

ARGX23: Invalid section number
ARGX24: Invalid count
SMAPX1: Attempt to delete a section still shared
SMAPX2: Indirect section map loop detected

Sets various flags and parameters in the monitor's data base. Most flag-oriented items are set by specifying 1 in AC2 and cleared by specifying 0 in AC2. In a few cases (noted in the text), flag-oriented items are set by setting and clearing the appropriate bit(s) in AC2. Value-oriented items are set to the value in AC2.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled. Some functions are for TCP/IP systems only.

ACCEPTS IN AC1: Function code

AC2: New value for the indicated function

RETURNS +1: Always

The codes for the functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .SFFAC | FACT file entries are allowed. |
| 1 | .SFCDE | CHECKD found errors. |
| 2 | .SFCDR | CHECKD is running. |
| 3 | .SFMST | Manual start is in progress. |
| 4 | .SFRMT | Remote LOGINs (dataset lines) are allowed. |
| 5 | .SFPTY | PTY LOGINs are allowed. |
| 6 | .SFCTY | CTY LOGINs are allowed. |

TOPS-20 MONITOR CALLS
(SMON)

| | | |
|----|--------|---|
| 7 | .SFOPR | Operator is in attendance. |
| 10 | .SFLCL | Local LOGINs (hardwired lines) are allowed. |
| 11 | .SFBTE | Bit table errors found on startup. |
| 12 | .SFCRD | Users can change nonprivileged directory parameters with the CRDIR monitor call. |
| 13 | .SFNVT | TCP/IP terminal LOGINs are allowed. |
| 14 | .SFWCT | WHEEL LOGINs on CTY are allowed. |
| 15 | .SFWLC | WHEEL LOGINs on local terminals are allowed. |
| 16 | .SFWRM | WHEEL LOGINs on remote terminals are allowed. |
| 17 | .SFWPT | WHEEL LOGINs on PTYs are allowed. |
| 20 | .SFWNV | WHEEL LOGINs on network virtual terminals (NVT) are allowed. |
| 21 | .SFUSG | USAGE file entries are allowed. |
| 22 | .SFFLO | Disk latency optimization using the RH20 backup register is enabled. This feature is not to be enabled unless the M8555 board of the RH20 is at Revision Level D AND either of the KL10-C processor is at Revision Level 10 or KL10-E processor is at Revision Level 2. |
| 23 | .SFMTA | If set, indicates that MOUNTR magtape allocation is enabled. |
| 24 | .SFMS0 | Set system message level 0 AC2: 1 (SF%MS0) to set; 0 to clear |
| 25 | .SFMS1 | Set system message level 1 AC2: 1 (SF%MS1) to set; 0 to clear |
| 26 | .SFBGS | Send operator messages to CTY; if off, such messages as BUGINF, BUGCHK, and "resource low" will be sent to OPR terminals, rather than the CTY. AC2: 1 (SF%BGS) to send to CTY; 0 to send to OPR |
| 27 | .SFMCB | Allow DECnet logins AC2: 1 (SF%MCB) to set; 0 to clear |
| 30 | .SFDPR | Enable disk preallocation. |
| 31 | .SFLAT | Enable LAT LOGINs. |
| 32 | .SFWLT | Enable WHEEL LOGINs on LAT terminals. |
| 44 | .SFNTN | Turn TCP/IP on. |
| 45 | .SFNDU | Reinitialize TCP/IP if it is down. |
| 46 | .SFNHI | Initialize TCP/IP host table. |
| 47 | .SFTMZ | Set the local time zone to the value given in AC2. |
| 50 | .SFLHN | Set the local TCP/IP host number to the value given in AC2. |
| 51 | .SFAVR | Account validation will be running on this system. |
| 52 | .SFSTS | Enable/disable status reporting. |
| 53 | .SFSOK | Set GETOK% defaults AC2: Flags,,GETOK% function code |

Bit Symbol Meaning

| | | |
|----|--------|-----------------------------|
| B0 | SF%EOK | 0 = Disable access checking |
| | | 1 = Enable access checking |

TOPS-20 MONITOR CALLS
(SMON)

B1 SF%DOK 0 = Deny access if checking disabled
1 = Allow access if checking disabled

This function should be given by the access-control program (supplied by the installation) to turn on access checking for each of the desired functions. It is also used to set the default action for each function that is not being checked by the access-control program. Installation-defined function codes (400000+n) must be enabled/disabled by using function code 400000, regardless of the installation-defined function code given in the GETOK% call. If there is no access-control program, the default action of the GETOK% JSYS will be to deny access for any installation-defined function code.

See the description of the GETOK% JSYS for GETOK% function codes.

| | | |
|----|--------|---|
| 54 | .SFMCY | Specifies the maximum offline expiration period (tape recycle period) in days, for ordinary files. |
| 55 | .SFRDU | Read date update function |
| 56 | .SFACY | Specifies the maximum offline expiration period (tape recycle period) in days, for archive files. |
| 57 | .SFRTW | Sets/clears the no-retrieval-waits flag in the monitor. When set, this specifies that those file retrievals requests that are waiting for the retrieval should fail rather than wait. |
| 60 | .SFTDF | Set tape mount controls |

Flags:

| Bit | Symbol | Meaning |
|-----|--------|--|
| B0 | MT%UUT | 1 unload unrecognizable tapes 0 treat unrecognizable tapes as unlabeled |

| | | |
|----|--------|--|
| 61 | .SFWSP | Enable working set preloading |
| 62 | .SFDST | Set Daylight Saving Time conversion method |

Value Symbol Meaning

| | | |
|---|--------|----------------------------------|
| 0 | .DSTAU | Perform automatic DST changeover |
| 1 | .DSTNV | Never perform DST changeover |
| 2 | .DSTAL | Always perform DST conversion |

| | | |
|----|--------|--|
| 63 | | Reserved for DIGITAL. |
| 64 | .SFMSD | Set MSCP access for disk drive; this function allows or restricts other systems' access to local MASSBUS disks on a per drive basis. |

TOPS-20 MONITOR CALLS
(SMON)

AC2 contains address of an argument block in the following format:

| Offset | Symbol | Meaning |
|--------|--------|---|
| 0 | .SVCNT | length of the block, including this word |
| 1 | .SVTYP | flags and drive type Flag: B0(MS%DDU) if set, the drive is RESTRICTED; if not set, the drive is ALLOWED. |
| 2 | .SVDSH | high order serial number of disk drive |
| 3 | .SVDSN | low order serial number of disk drive |

The following errors are possible on failure of this function:

MSCPX1: No MSCP server in current monitor
MSCPX2: Drive type error
MSCPX3: Requested drive not found
MSCPX4: MSCP server not currently running

| | | |
|----|---------|---|
| 65 | .SFSPR | Set SPEAR event counter |
| 66 | .SFCOT | Set time between carrier off event (including network connection being broken) and automatic logout of the job. AC2 is the time in milliseconds. The default is 5 minutes. |
| 67 | .SFHU0 | Control hang up action for jobs not logged in AC2: 0 to not hang up; 1 to hang up The default is to hang up. |
| 70 | .SFHU1 | Control hang up action for jobs logged in AC2: 0 to not hang up; 1 to hang up The default is to not hang up. |
| 71 | .SFEXEC | Flag word for configurations for the EXEC AC2 Flags: B0(XC%FST) do not allow /FAST option on LOGIN |
| 72 | .SFSEA | Set Ethernet address. AC2 contains the Ethernet interface channel number. AC3 contains a byte pointer to the 6 (8-bit) byte Ethernet address. |
| 73 | .SFDCCD | Set "don't care" disk. Used to indicate that a drive may be accessed without coordinating accesses with other processors. Arguments are the same as for the .SFMSD function, however, no flags are allowed. |

The following errors are possible on failure of this function:

TOPS-20 MONITOR CALLS
(SMON)

DIAGX9: Unit does not exist
MSTX14: Invalid channel number
MSTX15: Invalid unit number
MSTX16: Invalid controller number
MSTX27: Specified unit is not a disk
MSTX41: Channel does not exist
MSTX42: Controller does not exist

| | | |
|-----|---------|--|
| 74 | .SFLTS | Set Local Area Transport (LAT) state. AC2 contains the LAT state: LS.OFF for off, or LS.ON for on. |
| 75 | .SFCLU | Controls whether or not this system allows a remote INFO% to be performed on this system. AC2: 0 to allow remote INFO%, 1 to not allow remote INFO% (default is to allow, 0). |
| 76 | .SFTMG | Controls whether or not this system allows remote TTMSG% to be performed on this system. AC2: 0 to allow remote TTMSG%, 1 to not allow remote TTMSG% (default is to allow, 0). |
| 77 | .SFOFS | Set the offline structures timeout interval. Valid intervals are from 1 to 900 seconds; 0 disables offline structures. |
| 100 | .SFLGS | Enable the login structure feature. If disabled, the monitor doesn't search for a login structure at system startup. |
| 101 | .SFMPPL | Set minimum password length. AC2: Minimum length or 0 to disable. Minimum length must be 1 to 39 characters. |
| 102 | .SFACJ | This function only takes a valid argument of 0 in AC 2. This will start up an ACJ process in the monitor if one is not already running. The monitor will get the program to run from DEFAULT-ACJ:. If the DEFAULT-ACJ: logical name does not exist, the system will try to get the file from SYSTEM:ACJ.EXE. |
| 103 | .SFPEX | Controls password expiration. Sets a system wide parameter that is used to determine the expiration date and time when a user changes his password. For example, if a password was set on May 6, 1988 at 14:03 and the system had password expiration enabled for 10 days, then the password that was just set would expire on May 16, 1988 at 14:03. AC2: 0 - Disable password expiration, 1-366 - Number of days a password remains valid. |
| 104 | .SFPWD | Used to enable or disable the password dictionary feature. If enabled, words listed in SYSTEM:PASSWORD.DICTIONARY are not allowed as valid passwords. AC2: 0 - Disable password dictionary, 1 - Enable password dictionary |

TOPS-20 MONITOR CALLS
(SNDIM)

SNDIM ERROR MNEMONICS:

SNDIX1: Invalid message size
SNDIX2: Insufficient system resources (no buffers available)
SNDIX3: Illegal to specify NCP links 0-72
SNDIX4: Invalid header value for this queue
SNDIX5: IMP down
SQX1: Special network queue handle out of range
SQX2: Special network queue not assigned

Sends an Internet datagram. Internet queues are assigned by ASNIQ%.

RESTRICTIONS: For TCP/IP systems only.

ACCEPTS IN AC1: Internet queue handle

AC2: Address of message buffer

AC3: Not used, must be 0

RETURNS +1: Failure, with error code in AC1

+2: Success

The message buffer must contain the total word count for the buffer in word 0, a valid Internet header in B0-31 of words 1 through 5, and, optionally, data in words 6 through n.

If .IQPTM was nonzero in the ASNIQ% call (the queue was assigned with port-filtering turned on), then the port(s) must be in the word following the Internet header. The address of this word can be obtained by adding the address of word -1 in the buffer to the number in the Internet data offset field.

The monitor supplies the source host field and the Internet header checksum field in the header. The remainder of the header must be supplied by the caller.

SNDIN% ERROR MNEMONICS:

SNDIX1: Invalid message size
SNDIX2: Insufficient system resources (no buffers available)
SNDIX3: Illegal to specify NCP links 0-72
SNDIX4: Invalid header value for this queue

TOPS-20 MONITOR CALLS
(SNDIN%)

SNDIX5: IMP down
SQX1: Special network queue handle out of range
SQX2: Special network queue not assigned

Performs system performance analysis. The process can patch any instruction in the monitor with this call. For example, the user program can build a PC histogram by patching an instruction in the code for the 1.0-millisecond clock.

The general procedure for using the SNOOP call is as follows:

1. The user program supplies a set of breakpoint routines that are called by the monitor when control reaches one of the patched instructions. These routines are mapped into the monitor's address space into an area selected by the monitor. Thus, the routines must have self-relocating code or must be relocated by the user program to where they will be run, based on the monitor address supplied by the monitor.
2. The user program defines a number of breakpoints, analogous to DDT breakpoints.
3. The user program inserts all of the breakpoints simultaneously.
4. The user program goes to "sleep" or waits for terminal input while its breakpoint routines obtain control.
5. When the user program determines that the routines have completed, it removes the breakpoints.

The user program breakpoint routines run in the monitor address space, which means that the addresses of the code and the data are monitor addresses. The user program must modify these addresses, based on the values returned by the monitor, after the initialization but before the "snooping." The breakpoint routines must preserve any accumulators they use. Also, they must not cause a page fault if at interrupt level or if a patch has been made in the page fault handler or in the scheduler. Thus, the breakpoint routines should test for swappable code being in memory before referencing it. If swappable code needs to be referenced, the swappable monitor can be locked in memory, if desired. When a patch is made to a routine called at many interrupt levels, the program must specify a reentrant instruction to be used for patching.

TOPS-20 MONITOR CALLS
(SNOOP)

RESTRICTIONS: Requires enabled WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Function code

AC2: Function-specific argument

AC3: Function-specific argument

AC4: Function-specific argument

RETURNS +1: Failure, error code in AC1

+2: Success

The following functions are available:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .SNPLC | Declare and lock code into the monitor's address space. AC2: number of pages desired AC3: page number in user space of start of breakpoint routines to be locked On return, the pages are locked contiguously in the monitor's address space, and AC2 contains the monitor page numbers corresponding to the given user page number. |
| 1 | .SNPLS | Lock the swappable monitor. This function is useful for analyzing swappable data at interrupt level. On return, the entire swappable monitor is locked. |
| 2 | .SNPDB | Define a breakpoint AC2: number of breakpoint AC3: address in monitor space to be patched. The patched instruction can be a skip type instruction or a PUSHJ instruction, and the patching is similar to that in DDT. The routines will receive control before the patched instruction is executed. AC4: instruction to be executed before the patched instruction is executed. The instruction can be: |

TOPS-20 MONITOR CALLS
(SNOOP)

JSR LOC where LOC is an address in monitor space of the user's routine.

PUSHJ P,LOC when reentrant or recursive code is patched.

AOS LOC to count frequency of monitor execution points.

The error return is given if breakpoints have already been inserted.

NOTE

Putting a SNOOP breakpoint on a PUSHJ or other subroutine call instruction (including JSYS, MDISMS, etc) can cause problems. If the process is not in a NOSKED state already, it can be rescheduled during the breakpoint, in which case the breakpoint is removed, and the subsequent return is made to non-existent code.

- 3 .SNPIB Insert all breakpoints and start analyzing.
- 4 .SNPRB Remove all breakpoints and stop analyzing.
- 5 .SNPUL Unlock and release all storage, and undefine and remove all breakpoints.
- 6 .SNPSY Obtain the address of a monitor symbol.

AC2: radix-50 symbol

AC3: radix-50 program name if a local address is desired. If AC3 is 0, the entire symbol table is searched.

On return, AC2 contains the monitor address or value of the symbol.

- 7 .SNPAD Obtain a monitor symbol. (Requires MAINTENANCE capability)

AC2: 36-bit value of symbol that is to be looked up in the monitor's symbol table.

TOPS-20 MONITOR CALLS
(SNOOP)

AC3: radix-50 program name if a local value is desired. If AC3 is 0, the entire symbol table is searched.

On return, AC2 contains the first radix-50 monitor symbol that is closest to and has a value less than the specified value, and AC3 contains the difference between the value of the symbol returned and the specified value.

SNOOP ERROR MNEMONICS:

SNOPX1: WHEEL or OPERATOR capability required
SNOPX2: Invalid function
SNOPX3: .SNPLC function must be first
SNOPX4: Only one .SNPLC function allowed
SNOPX5: Invalid page number
SNOPX6: Invalid number of pages to lock
SNOPX7: Illegal to define breakpoints after inserting them
SNOPX8: Breakpoint is not set on instruction
SNOPX9: No more breakpoints allowed
SNOP10: Breakpoints already inserted
SNOP11: Breakpoints not inserted
SNOP12: Invalid format for program name symbol
SNOP13: No such program name symbol
SNOP14: No such symbol
SNOP15: Not enough free pages for snooping
SNOP16: Multiply-defined symbol
SNOP17: Breakpoint already defined
SNOP18: Data page is not private or copy-or-write

Tests to see if the designated file output buffer is empty.

ACCEPTS IN AC1: Destination designator

RETURNS +1: Output buffer is not empty. AC2 contains the number of bytes remaining in output buffer, or 0 if output is in progress.

 +2: Output buffer is empty; AC2 contains 0. This return is given if an error occurs on the call; AC2 contains the appropriate error code.

TOPS-20 MONITOR CALLS
(SOBE)

If the designator is not associated with a terminal, the +2 return is given.

The SIBE call can be used to determine if the input buffer is empty.

SOBE ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Tests to see if the designated file output buffer is full.

ACCEPTS IN AC1: File designator

RETURNS +1: Output buffer is not full. This return is given if an error occurs on the call; AC2 will contain 0.

+2: Output buffer is full

On either return, the number of bytes remaining in the output buffer is returned in AC2 (if no error occurred on the call).

Writes a string from the caller's address space to the specified destination. The string can be a specified number of bytes or terminated with a specified byte.

ACCEPTS IN AC1: Destination designator

AC2: Byte pointer to string to be written

AC3: Count of the number of bytes in string, or 0

TOPS-20 MONITOR CALLS
(SOUT)

AC4: Byte (right-justified) on which to terminate output

RETURNS +1: Always, with updated string pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 controls the number of bytes to write.

AC3=0 The string being written is terminated with a 0 byte.

AC3>0 A string of the specified number of bytes is to be written or a string terminated with the byte given in AC4 is to be written, whichever occurs first.

AC3<0 A string of minus the specified number of bytes is to be written.

The contents of AC4 is ignored unless the contents of AC3 is a positive number.

If AC3 is a negative number and the destination designator refers to memory, then the string being written is terminated with a 0 byte. The byte pointer is left positioned before this 0 byte.

The output is terminated when the byte count becomes 0, the specified terminating byte is reached, or an error occurs during the transfer. The specified terminating byte is copied to the destination.

After execution of the call, the file's pointer is updated for subsequent I/O to the file. AC2 is updated to point to the last byte written or, if AC3 contained 0, the last nonzero byte written. The count in AC3 is updated toward zero by subtracting the number of bytes written from the number of bytes requested to be written.

When the SOUT call is used to write data to a magnetic tape, it sends a series of bytes packed into records of the specified record size. The size of the records to write is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) words. Thus, if the record size is 1000 bytes, two SOUT calls, each writing 500 bytes, would write one record. If during the writing, the end of tape mark was passed, an error (IOX5) is given. However, the data has been successfully written and the device status word has the MT%EOT bit set to indicate this condition. See Section 2.4.7 for more information about magnetic tape I/O.

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(SOUT)

SOUT ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX5: File is not open
IOX2: File is not opened for writing
IOX5: Device or data error
IOX6: Illegal to write beyond absolute end of file
IOX7: Insufficient system resources (Job Storage Block full)
IOX8: Monitor internal error
IOX11: Quota exceeded
IOX33: TTY input buffer full
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Writes a variable-length record from the caller's address space to the specified device.

If the record is to be written to magnetic tape, the maximum size of the record to write is specified with either the SET TAPE RECORD-LENGTH command or the .MOSRS function of the MTOPR call. The default record size is 1000(octal) bytes.

ACCEPTS IN AC1: Destination designator
AC2: Byte pointer to string to be written
AC3: Count of number of bytes in string, or 0
AC4: Byte (right-justified) on which to terminate output (optional)

RETURNS +1: Always, with updated byte pointers in AC2 and AC1, if pertinent, and updated count in AC3, if pertinent

The contents of AC3 and AC4 are interpreted in the same manner as they are in the SOUT monitor call.

Each SOUTR call writes at least one record. Thus, the caller can write variable-length records by indicating in AC3 the number of bytes to write in the record. If the SOUTR call requests more bytes to be written than the maximum record size, then records of the maximum size

TOPS-20 MONITOR CALLS
(SOUTR)

are written, plus another record containing the remaining bytes. If the SOUTR call requests fewer bytes than the maximum, or a number equal to the maximum, to be written, then records of the requested size are written.

The SOUTR call differs from the SOUT call in that the SOUTR call writes records on the tape upon execution of the call. The SOUT call does not write a record on the tape until the number of bytes equal to the record size have been written. Thus, if a record is being made from several strings in the caller's address space, the SOUT call can be used for the first strings and the SOUTR call for the last string.

For a TCP/IP transmission, SOUTR will set the TCP PUSH flag for the last message generated by the call and force all data held in local buffers to be sent immediately.

Can cause several software interrupts or process terminations on certain file conditions. (See bit OF%HER of the OPENF call description.)

Generates an illegal instruction interrupt on error conditions below.

SOUTR ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
IOX2: File is not open for writing
IOX5: Device or data error
IOX6: Illegal to write beyond absolute end of file
IOX7: Insufficient system resources (Job Storage Block full)
IOX8: Monitor internal error
IOX9: Function legal for sequential write only
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Sets the accessibility of a page. This call affects the map word of the page named in AC1 (no indirect pointers are allowed).

ACCEPTS IN AC1: Process/file designator in the left half, and page number within the file or process in the right half

TOPS-20 MONITOR CALLS
(SPACS)

AC2: Access information

B2(PA%RD) Permit read access

B3(PA%WT) Permit write access

B4(PA%EX) Permit execute access

B9(PA%CPY) Copy-on-write

RETURNS +1: Always

When used to modify a process page, the SPACS call does not allow any greater access than can be obtained with the PMAP call (that is, the access specified on the OPENF call is applied to SPACS operations involving file pointers).

The SPACS call does not allow bits to be set in a page that does not already exist.

The RPACS monitor call can be used to obtain the accessibility of a page.

Generates an illegal instruction interrupt on error conditions below.

SPACS ERROR MNEMONICS:

DESX1: Invalid source/destination designator

DESX2: Terminal is not available to this job

DESX3: JFN is not assigned

DESX4: Invalid use of terminal designator or string pointer

DESX5: File is not open

DESX8: File is not on disk

SPACX1: Invalid access requested

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

FRKHX3: Invalid use of multiple process handle

FRKHX8: Illegal to manipulate an execute-only process

Sets the primary JFNs (.PRIIN and .PRIOU) for the specified process.

ACCEPTS IN AC1: Process handle

TOPS-20 MONITOR CALLS
(SPJFN)

AC2: Primary input JFN in the left half, and primary output JFN in the right half

RETURNS +1: Always

The JFNs given cannot be either 100 or 101. These JFNs cause the specified process to receive an error on any primary I/O operation. If minus one is placed in the appropriate half of AC2, the primary input/output JFNs are set to the process's controlling terminal.

The GPJFN monitor call can be used to obtain the primary JFNs.

Generates an illegal instruction interrupt on error conditions below.

SPJFN ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
DESX3: JFN is not assigned

Changes (splices) the process structure of a job. This monitor call allows two types of changes to the process structure. The first type allows two parallel processes to be spliced such that one process becomes the superior of the other. The second type permits a process to splice its inferior to its superior, thereby deleting the calling process. The paragraphs below describe the calling sequences for the two types.

Case I - Inserting a process between a given process and one of its inferiors

In this case, the new process structure provides superior process capabilities that were not available between parallel processes. The process that becomes the new superior must be either the one executing the SPLFK call or an inferior of it. The new superior process must not be the same as the new inferior process, and must not be inferior to the new inferior process. The new inferior and all of its inferiors will be frozen after execution of the SPLFK call.

ACCEPTS IN AC1: Process handle of the new superior process

TOPS-20 MONITOR CALLS
(SPLFK)

AC2: Process handle of the new inferior process

RETURNS +1: Failure, error code in AC1

 +2: Success, a process handle in AC1. This handle may be used by the new superior process (in AC1) to refer to its new inferior (in AC2).

Case II - Removing a process as the superior of another process

In this case, the new process structure allows a process to begin or continue execution as a logical replacement of the calling process. The calling process can splice only one inferior in place of itself. After the execution of the call, the calling process is halted, its process's pages are unmapped, it is removed from the process structure, and it is completely replaced by the inferior process. Any other inferiors of the calling process are removed as well. In other words, the calling process and its remaining inferiors will be treated as if the process had been removed with the KFORK% monitor call. The process that is spliced to the calling process's superior uses the process handle of the calling process and continues with any functions that were being performed by the superior before the execution of the SPLFK% call.

ACCEPTS IN AC1: B0(SF%EXT) and the address of an argument block in the following format:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .SFLEN | Length of argument block including this word |
| 1 | .SFCOD | Function code. Currently, only the function .SFUNS (code 1) is defined to remove a process and continue or start the new inferior. |
| 2 | .SFUIN | Process handle of the new inferior process |
| 3 | .SFUFL | Flags |
| 4 | .SFUA1 | PC flags,,0 or entry vector offset (see description of flag bits below) |
| 5 | .SFUA2 | Starting address if SF%ADR is set |

The flag bits in word .SFUFL are as follows:

| Bit | Symbol | Meaning |
|-----|--------|---|
| 0 | SF%CON | continue the new inferior from where it was halted. If SF%CON is set, the address in word |

TOPS-20 MONITOR CALLS
(SPLFK)

.SFUA1 is ignored, and the process continues from where it was halted.

- 1 SF%GO start the new inferior at the entry vector offset in word .SFUA1.
- 2 SF%ADR interpret the contents of words .SFUA1 and .SFUA2 as flags and an address to start the new inferior process. If this flag is not set, the contents of word .SFUA1 are interpreted as an entry vector offset.

RETURNS +1: Failure, error code in AC1

 +2: Success, a process handle in AC1.

SPLFK ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX5: Process has not been started
FRKHX8: Illegal to manipulate an execute-only process
SFRVX1: Invalid position in entry vector
SPLFX1: Process is not inferior or equal to self
SPLFX2: Process is not inferior to self
SPLFX3: New superior process is inferior to intended inferior

Defines and initializes a device to be used for input spooling or sets and reads the directory for a spooled device.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Length of argument block in the left half, and function code in the right half

 AC2: Address of argument block

RETURNS +1: Failure, error code in AC1

 +2: Success

The format of the argument block is different depending upon the

TOPS-20 MONITOR CALLS
(SPOOL)

particular function desired. The available functions, along with their argument block formats, are as follows:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .SPLDI | Define an input spooling device. The argument block is: |
| | | Word Symbol Meaning |
| | 0 | .SPLDV Device designator of input device. |
| | 1 | .SPLNA Pointer to name string comprising the set of files to be input. |
| | 2 | .SPLGN Generation number of first file. This number is incremented by 1 each time the spooled device is opened. |
| 1 | .SPLSD | Set the directory of the spooled device. The argument block is: |
| | | Word Symbol Meaning |
| | 0 | .SPLDV Device designator of spooled device. |
| | 1 | .SPLDR Directory number. This number is the logged-in directory number of the user who opened the spooled device. |
| | | This function requires the process to have WHEEL or OPERATOR capability enabled. |
| 2 | .SPLRD | Read the directory of the spooled device. The argument block is: |
| | | Word Symbol Meaning |
| | 0 | .SPLDV Designator of spooled device. |
| | | The directory number of the spooled device is returned in word 1 of the argument block. |

To read from a spooled input device, the user first defines the name of the files comprising his set of spooled input files. The files have names in the format:

STR:<SPOOLED-DIRECTORY>DEVICE-DIR#.NAME.1,2,3,...

The spooled directory is the directory to receive any spooled input

TOPS-20 MONITOR CALLS
(SPOOL)

from the device. The .SPLSD function can be used by a privileged process to set the directory. The default directory for all of the spooled devices is <SPOOL>.

The device is the name of the device being used for spooled input. It is the same name that was given on the original GTJFN call.

The directory number is the logged-in directory number of the user that opened the spooled device.

The name is the name of the set of files to be input. The .SPLDI function is used to define this name.

The generation number begins with the value specified by the .SPLDI function and increments by one each time the spooled device is opened.

Thus, if the input spooler for the card reader (CDR) is reading files for a user whose directory number is 23, then the files might have names like the following:

<SPOOL>CDR-23.BATCH-SEQUENCE-37.1,2,3,...

To initialize the spooled card reader, the user would then execute the SPOOL call giving "BATCH-SEQUENCE-37" as the name of the set of files to be input and "1" as the beginning generation number.

SPOOL ERROR MNEMONICS:

SPLX1: Invalid function
SPLX2: Argument block too small
SPLX3: Invalid device designator
SPLX4: WHEEL or OPERATOR capability required
SPLX5: Illegal to specify 0 as generation number for first file
SPLX6: No directory to write spooled files into

Sets the priority word for the specified process.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Process handle

AC2: Priority word

TOPS-20 MONITOR CALLS
(SPRIW)

RETURNS +1: Always

See the SJPRI monitor call description for the format of the priority word.

Generates an illegal instruction interrupt on error conditions below.

SPRIW ERROR MNEMONICS:

WHELX1: WHEEL or OPERATOR capability required

Creates a sharable, save-format file for the given JFN by copying (not sharing) pages from the given process. (See Section 2.8.2 for the format of a sharable save file.) This monitor call is used for creating programs that can be shared. It saves the file in groups of contiguous pages for which the same access is desired. It always saves the entry vector, but saves only PDV addresses that are within the range of saved pages. (See PDVOP% description.) SSAVE closes and releases the given JFN.

ACCEPTS IN AC1: Process handle in the left half, and JFN in the right half

AC2: One table entry, or 0 in the left half and the address of the table in the right half (see below)

AC3: Second word of two-word table entry (if bit SS%EPN is set in AC2), or 0

RETURNS +1: Always

If the pages to be saved are all in section zero, the table has a one-word entry for each group of pages.

If any of the groups of pages to be saved is in a nonzero section, the table entry for that group is two words long (see below). Bit SS%EPN must be set in the first word, and bits 27-35 are zero in the first word. The second word contains the number of the first page in the group (right-justified).

A zero word ends the table.

The first word of each table entry has the following format:

TOPS-20 MONITOR CALLS
(SSAVE)

| Bit | Symbol | Meaning |
|-------|--------|--|
| 0-17 | SS%NNP | Negative of the number of pages in each group (right-justified). |
| 18 | SS%CPY | Allow copy-on-write access to the group of pages. |
| 19 | SS%UCA | Limit the access according to the current access of the user's page. (See below.) |
| 20 | SS%RD | Allow read access to the group of pages. |
| 21 | SS%WR | Allow write access to the group of pages. |
| 22 | SS%EXE | Allow execute access to the group of pages. |
| 23 | SS%EPN | Each table entry is two words long, and the second word contains the page number of the first page of each group. |
| 27-35 | SS%FPN | If SS%EPN is not set, this field contains the number of the first page in the group (right-justified). If SS%EPN is set, this field is zero, and the number of the first page in the group is in word two of this table entry. |

When B19(SS%UCA) is set, the access to the group of pages is determined by ANDing the access bits specified in the table word with the corresponding access bits for the user's pages (as determined by the RPACS call). This means that a given access is allowed only if both the SSAVE call indicates it and the page currently has it. If B19(SS%UCA) is not set, the access granted to the group of pages is that indicated by the bits set in the table word.

The SSAVE call does not save the accumulators nor does it save nonexistent pages.

The GET monitor call is used to map a file saved with the SSAVE call back into a given process.

Can cause several software interrupts or process terminations on certain file conditions.

Generates an illegal instruction interrupt on error conditions below.

SSAVE ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle
SSAV1: Illegal to save files on this device

TOPS-20 MONITOR CALLS
(SSAVE)

SSAVX2: Page count (left half of table entry) must be negative
SSAVX3: Insufficient system resources (Job Storage Block full)
SSAVX4: Directory area of EXE file is more than one page
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

All I/O errors can also occur.

Sets the system's date. (See Section 2.9.2.)

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Day in the left half, and fraction of the day in the right half

RETURNS +1: Failure, error code in AC1

+2: Success

The STAD call requires the process to have WHEEL or OPERATOR capability enabled if the system's date is already set.

The GTAD monitor call can be used to obtain the system's date.

STAD ERROR MNEMONICS:

STADX1: WHEEL or OPERATOR capability required

STADX2: Invalid date or time

Compares two ASCIZ strings in the caller's address space. Note that letters are always considered as upper case, regardless of their case within the string. Therefore, the strings ABC and abc are considered an exact match.

TOPS-20 MONITOR CALLS
(STCMP)

ACCEPTS IN AC1: Byte pointer to test string

AC2: Byte pointer to base string

RETURNS +1: Always, with

AC1 containing the compare code:

B0(SC%LSS) Test string is less than base string.

B1(SC%SUB) Test string is a subset of base string.

B2(SC%GTR) Test string is greater than base string.

AC2 containing base byte pointer, updated such that an ILDB instruction will reference the first nonmatching byte.

One string is considered less than another string if the ASCII value of the first nonmatching character in the first string is less than the ASCII value of the character in the same position in the second string.

One string is considered a subset of another string if both of the following conditions are true:

1. From left to right, the ASCII values of the characters in corresponding positions are the same.
2. The test string is shorter than the base string.

Two strings are considered equal if the ASCII values of the characters in corresponding positions are the same and the two strings are the same size. In this case, the contents of AC1 is 0 on return.

Translates the given device name string to its corresponding device designator.

ACCEPTS IN AC1: Byte pointer to the string to be translated

RETURNS +1: Failure, error code in AC2

TOPS-20 MONITOR CALLS
(STDEV)

+2: Success, device designator (see Section 2.4) in AC2

The string to be translated is terminated by the first space (ASCII code 40), null (ASCII code 0), or colon (ASCII code 72).

The DEVST monitor call can be used to translate a device designator to its corresponding string.

STDEV ERROR MNEMONICS:

STDVX1: No such device

Simulates terminal input.

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: File designator (only terminal designators are legal)

AC2: Character to be input, right-justified

RETURNS +1: Always

The character is taken from the accumulator and placed into the specified terminal's input buffer whether or not the buffer is empty. The DIBE call can be used to prevent sending an interrupt character (for example, CTRL/C) before the program has processed all of the previous input.

The STI monitor call requires the process to have WHEEL or OPERATOR capability enabled if the specified terminal either is not assigned or opened by the process or is not accepting advice. (See the TLINK bit TT%AAD.)

The use of this monitor call is not recommended for pseudo-terminals (PTYs). The recommended procedure for placing a character in the PTY input buffer is to open the PTY for output with OPENF and then perform output with the BOUT call.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(STI)

STI ERROR MNEMONICS:

TTYX1: Device is not a terminal
DESX2: Terminal is not available to this job
DEVX2: Device already assigned to another job
WHELX1: WHEEL or OPERATOR capability required
TTYX01: Line is not active

Sets the terminal interrupt word (see Section 2.6.6) for the entire job or a specific process. This call declares that terminal characters that usually cause an interrupt are instead to be passed to the program as input. In actuality, the STIW call sets the interrupt word mask, thus determining for each of the 36 terminal codes if the job or process should receive an interrupt. The call's effect is different, depending on whether the call is being executed for the entire job or for a specific process in the job.

When the STIW call is executed for the entire job, codes corresponding to the bits on in the mask will cause an interrupt if a process in the job has enabled for an interrupt on that code. If multiple processes have enabled that code, the lowest inferior process receives the interrupt. (If several processes at the same lowest level have enabled the code, the process that receives the interrupt is determined by the system.) If no process has enabled that code, the character corresponding to the code is passed to the program. Also, characters are passed to the program when their corresponding bits are off in the mask, even if a process has enabled that code. Initially, all codes are declared to cause an interrupt (that is, all bits in the mask are on), and the program can execute the RTIW call to determine the current status. Thus if the program wishes to read a terminal interrupt character as input, it executes the STIW call for the entire job and turns off the mask bit corresponding to the character.

When the STIW call is executed for a specific process in the job, codes corresponding to the bits on in the mask are assumed to be enabled by the specific process and cause an interrupt if in fact they are enabled. If the process has not enabled for the code, the character corresponding to the code is ignored, if it is typed. Characters corresponding to the bits off in the mask are assumed not to be enabled by the process. This use of the STIW call is implicitly executed on an ATI call.

Each time the STIW call is executed for a specific process, the mask is changed to reflect the bits changed in that process.

TOPS-20 MONITOR CALLS
(STIW)

The STIW call sets or clears specific terminal codes for a particular process without actually changing the channel assignment that each code has. The ATI call is used to set the channel assignment, and the DTI call is used to clear the assignment.

The STIW call requires the process to have SC%CTC capability enabled to give -5 as an argument.

ACCEPTS IN AC1: B0(ST%DIM) Set the deferred terminal interrupt mask given in AC3

B18-B35 Process handle, or -5 for entire job
(ST%PRH)

AC2: Terminal interrupt word mask
Bit n on means terminal code n is enabled.

AC3: Deferred terminal interrupt word mask
Bit n on means terminal code n is deferred.

RETURNS +1: Always

The argument in AC3 is ignored, and no change is made to the deferred interrupt word mask, if B0(ST%DIM) is not set or if the process handle in AC1 does not indicate a specific process.

If multiple processes enable the same interrupt character and any one of the processes declares it deferred, the character is deferred for all the processes that enabled it.

The RTIW call can be used to obtain the terminal interrupt word masks.

STIW ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

Simulates terminal output.

ACCEPTS IN AC1: File designator (only terminal designators are legal)

TOPS-20 MONITOR CALLS
(STO)

RETURNS +1: Always, with the character right-justified in AC2

The character is taken from the specified terminal's output buffer and placed in the accumulator. The process is blocked until the character is in the accumulator.

The use of this monitor call is not recommended for pseudo-terminals (PTYs). The recommended procedure for reading a character from the PTY output buffer is to open the PTY for input with OPENF and then perform input with the BIN call.

STO ERROR MNEMONICS:

TTYX1: Device is not a terminal
DESX2: Terminal is not available to this job
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Sets the device-related modes for the specified terminal. The modes that can be set by this call are in the following bits of the JFN mode word. (See Section 2.4.9.1.)

| | |
|-----------------|--------------------------|
| B1(TT%MFF) | mechanical form feed |
| B2(TT%TAB) | mechanical tab |
| B3(TT%LCA) | lower case |
| B4-B10(TT%LEN) | page length |
| B11-B17(TT%WID) | page width |
| B25(TT%ECM) | echo control |
| B30(TT%UOC) | uppercase output control |
| B31(TT%LIC) | lowercase input control |
| B32-B33(TT%DUM) | duplex mode |
| B34(TT%PGM) | output page mode |

ACCEPTS IN AC1: File designator

AC2: JFN mode word

RETURNS +1: Always

The STPAR monitor call is a no-op if the designator is not associated with a terminal.

TOPS-20 MONITOR CALLS
(STPAR)

The SFMOD monitor call can be used to set program-related modes of the JFN mode word, and the RFMOD monitor call can be used to obtain the JFN mode word.

When the page length and width fields are set with the STPAR call, they have a maximum range of 127. The MTOPR call can be used to set these fields to values greater than 127. A nonzero value of less than 2 for the length or less than 10 for the width causes STPAR to leave the field unchanged.

STPAR ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX3: JFN is not assigned
DESX5: File is not open
DEVX2: Device already assigned to another job
TTYX01: Line is not active

Translates the given directory name string to its corresponding project-programmer number (a TOPS-10 36-bit directory designator). This project-programmer number is associated with the structure containing the given directory and is valid only for the current mounting of that structure. The STPPN monitor call and the PPNST monitor call should appear only in programs that require translations of project-programmer numbers. Both calls are temporary calls and may not be defined in future releases.

RESTRICTIONS: When this call is used in any section other than section zero, one-word global byte pointers used as arguments must have a byte size of seven bits.

ACCEPTS IN AC1: Byte pointer to ASCII string containing the directory name, a JFN, or a 36-bit directory number

RETURNS +1: Always, with the corresponding project-programmer number in AC2

STPPN ERROR MNEMONICS:

STRX02: Insufficient system resources
STRX03: No such directory name
STRX04: Ambiguous directory specification
DESX1: Invalid source/destination designator

TOPS-20 MONITOR CALLS
(STPPN)

DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DESX8: File is not on disk
DESX10: structure is dismantled

Clears the status of a file. (See the GTSTS monitor call for the format of the JFN status word.)

ACCEPTS IN AC1: JFN in the right half

AC2: STSTS flags. If a given STSTS flag is zero, then the associated flag in the JFN status word is cleared. If a given STSTS flag is one, no action is performed. Any undocumented bits in AC2 are ignored.

RETURNS +1: Failure, error code in AC1
+2: Success

The STSTS call is used to clear the following bits of the status word:

B9(GS%ERR) file may be in error
B13(GS%HLT) I/O errors are terminating conditions (set by OPENF)
B17(GS%FRK) this is a restricted JFN. Only the process that received it may use it. Other processes may reference the file with other JFNs. (Set by GTJFN)

STSTS ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer

TOPS-20 MONITOR CALLS
(STTYP)

Sets the terminal type number for the specified terminal line. (See Section 2.4.9.4.)

ACCEPTS IN AC1: Terminal designator
 AC2: Terminal type number

RETURNS +1: Always

The STTYP call sets the bits in the JFN mode word for mechanical form feed and tab, lower case, and page length and width according to their settings in the device characteristics word. These bits can subsequently be changed with the STPAR monitor call.

The GTTYP monitor call can be used to obtain the terminal type number for a specified line.

STTYP ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
STYPX1: Invalid terminal type
TTYX01: Line is not active

Swaps the association of two JFNs by literally exchanging all information cells of each JFN.

ACCEPTS IN AC1: JFN
 AC2: Another JFN

RETURNS +1: Always

SWJFN ERROR MNEMONICS:

DESX1: Invalid source/destination designator
DESX2: Terminal is not available to this job
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
SWJFX1: Illegal to swap same JFN

TOPS-20 MONITOR CALLS
(SWTRP%)

Provides a process with the ability to intercept arithmetic overflow or underflow conditions efficiently. Use of the SWTRP% JSYS to trap for these conditions is more efficient in some applications than using the software interrupt system.

SWTRP% also allows a process to declare its LUUO block for LUUOs executed in nonzero sections.

ACCEPTS IN AC1: Process handle

AC2: Function code

AC3: Function-dependent argument

RETURNS +1: Always

The functions are as follows:

| Code | Symbol | Function |
|------|--------|---|
| 0 | .SWART | Set arithmetic trap location AC3 contains the address of the arithmetic trap block (see LUUO block below). A zero in AC3 clears the arithmetic trap. |
| 1 | .SWRAT | Read arithmetic trap location Returns the trap block address in AC3 (see LUUO block below). A zero is returned if an arithmetic trap is not set. |
| 2 | .SWLUT | Set LUUO block address for nonzero sections AC3 contains the address. A zero in AC3 clears the location. See below for the format of the LUUO block. |
| 3 | .SWRLT | Read LUUO block address Returns the address in AC3. A zero is returned if no block is currently in effect. |

The LUUO block has the following format:

TOPS-20 MONITOR CALLS
(SWTRP%)

| Offset | 0 | 12 | 13 | 17 | 18 | 26 | 27 | 30 | 31 | 35 |
|-----------|------------------------------------|----|----|----|----|----|----|----|----|----|
| | ===== | | | | | | | | | |
| .ARPFL(0) | ! PC flags ! 0 ! opcode ! AC ! 0 ! | | | | | | | | | |
| | ----- | | | | | | | | | |
| .AROPC(1) | ! 0 ! Location of LUUO +1 ! | | | | | | | | | |
| | ----- | | | | | | | | | |
| .AREFA(2) | ! 0 ! E of the LUUO ! | | | | | | | | | |
| | ----- | | | | | | | | | |
| .ARNPC(3) | ! 0 ! New PC ! | | | | | | | | | |
| | ===== | | | | | | | | | |
| | 0 | 5 | 6 | | | | | | | 35 |

4 .SWSPD Set PDL overflow trap

5 .SWRPD Read PDL overflow trap

An LUUO executed in section zero will store the opcode, AC, and effective address of the LUUO in user location 40, and will execute the instruction in user location 41. An LUUO executed in a nonzero section makes use of the UPT (user process table). SWTRP% allows a process to store the desired address in the UPT so that subsequent LUUOs will produce the desired effect. The address in the UPT points to the LUUO block shown above. This block is stored in the user's address space). See the Processor Reference Manual for more information on LUUOs.

Places information in the System Error file (ERROR.SYS). (See the SPEAR Manual for information on the system error file, <SYSTEM-ERROR>ERROR.SYS.)

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Address of argument block

AC2: Length of argument block

RETURNS +1: Always

The first four words of the header block must contain the standard header information required by SPEAR.

Generates an illegal instruction interrupt on error conditions below.

TOPS-20 MONITOR CALLS
(SYERR)

SYERR ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required
SYEX1: Unreasonable SYSERR block size
SYEX2: No buffer space available for SYSERR

Returns the table number, table length, and word 0 of the specified system table. (See Section 2.3.2 for the names of the system tables.)

ACCEPTS IN AC1: SIXBIT table name

RETURNS +1: Always, with

AC1 containing word 0 of the table

AC2 containing the negative of the number of words in
the table in the left half, and the table number
in the right half

The table number returned can be given to the GETAB monitor call as an argument. However, because the MONSYM file includes symbol definitions for the system tables, execution of the SYSGT call is not required to obtain the table number for the GETAB call.

The contents of AC2 is 0 on return if the specified table was not found.

Adds an entry to a standard-formatted command table used for user program command recognition. (See the TBLUK call description for the format of the command table.)

ACCEPTS IN AC1: Flag bits in the left half, and address of word 0
(header word) of table in the right half

B0(TB%ABR) Abbreviations are present in keyword
table

TOPS-20 MONITOR CALLS
(TBADD)

AC2: Entry to be added to table (see the TBLUK call for the format of a table entry)

RETURNS +1: Always, with address in the table of the new entry in AC1

Generates an illegal instruction interrupt on error conditions below.

TBADD ERROR MNEMONICS:

TADDX1: Table is full
TADDX2: Entry is already in table

Deletes an entry from a standard-formatted command table used for user program command recognition. (See the TBLUK call description for the format of the command table.)

ACCEPTS IN AC1: Flag bits in the left half, and address of word 0 (header word) of table in the right half

B0(TB%ABR) Abbreviations are present in keyword table

AC2: Address of entry to be deleted; this address is returned in AC1 on a TBLUK call

RETURNS +1: Always

Generates an illegal instruction interrupt on error conditions below.

TBDEL ERROR MNEMONICS:

TDELX1: Table is empty
TDELX2: Invalid table entry location

TOPS-20 MONITOR CALLS
(TBLUK)

Compares the specified string in the caller's address space with strings indicated by a command table. The table has a standard format, which is described below.

This call is used to implement a consistent style of command recognition and command abbreviation for user programs. The TBLUK call performs the function of string lookup in the table, and the TBADD and TBDEL calls perform the functions of adding to and deleting from the table.

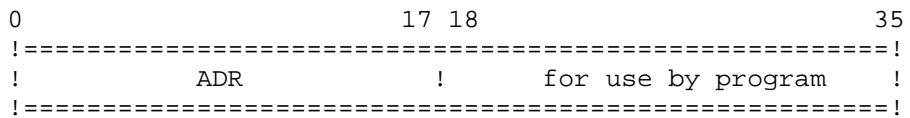
The command table has the following format:

| Word | Meaning |
|-------------|--|
| 0 | Number of entries in the table (not including this entry) in the left half, and maximum number of entries in the table (not including this entry) in the right half. |
| 1 through n | Address of an argument block in the left half; the right half of each table entry is available for use by the user program. |

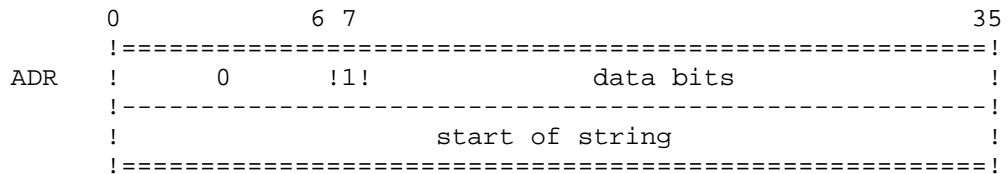
The argument block can have one of two formats. Bits 0-7 of the first word of the argument block determine which format the argument block has.

If bits 0-6 are all off and B7(CM%FW) is on, the string begins in the next word of the argument block, and the remainder of this word contains data bits relevant to the string.

Table Entry



Argument Block



The following bits are currently defined:

TOPS-20 MONITOR CALLS
(TBLUK)

| Bit | Symbol | Meaning |
|-----|--------|--|
| 34 | CM%NOR | Do not recognize this string, even if a string is specified that matches exactly, and consider an exact match as ambiguous. A program can set this bit to include entries that are initial substrings of other entries in the table to enforce a minimum abbreviation of these other entries (for example, to include D and DE in the table to enforce DEL as the minimum abbreviation of DELETE). |
| 7 | CM%FW | Indicate that the remainder of this word is a flag word containing data bits relevant to the string. This bit must be on to distinguish a flag word from a null string. |

If any bit of bits 0-6 of the first word of the argument block is on or if B7(CM%FW) is off, the string begins in that word. In this case, the data bits do not apply and are assumed to be off.

Table Entry

```

0                               17 18                               35
!=====!
!           ADR           !           !           !
!=====!

```

Argument

```

0                               35
!=====!
ADR !           start of string           !
!=====!

```

The addresses in the command table must be sorted according to the alphabetical order of the strings. Note that letters are always considered as uppercase. Therefore, the strings ABC and abc are considered equivalent strings. This order results in efficient searching of strings and determination of ambiguous strings.

The right half of each table entry can be used by the program for an address to a dispatch table for the command or for a pointer to a parameter block for additional information about the call. The contents of this half word is ignored by the three table calls.

- ACCEPTS IN AC1: Address of word 0 (header word) of table
- AC2: Byte pointer to string in caller's address space that is to be compared with the string in the table
- RETURNS +1: Always, with

TOPS-20 MONITOR CALLS
(TBLUK)

AC1 containing the address of the entry that matches the input string or address where the entry would be if it were in the table.

AC2 containing recognition bits:

B0(TL%NOM) The input string does not match any string in the table.

B1(TL%AMB) The input string matches more than one string in the table (that is, it is ambiguous).

B2(TL%ABR) The input string is a valid abbreviation of a string in the table.

B3(TL%EXM) The input string is an exact match with a string in the table.

AC3 containing a byte pointer to the remainder of the string in the table if the match was on an abbreviation (TL%ABR is on). This string can then be output to complete the command.

Generates an illegal instruction interrupt on error conditions below.

TBLUK ERROR MNEMONICS:

TLUKX1: Internal format of table is incorrect

Provides Internet terminal control protocol operations.

RESTRICTIONS: Requires WHEEL, MAINTENANCE, or NET WIZARD capability; for TCP/IP systems only.

ACCEPTS IN AC1: JFN of connection

AC2: Function code

AC3: Function argument or address of argument block

AC4: Function-specific argument

RETURNS: +1 Always

TCOPR% Functions:

TOPS-20 MONITOR CALLS
(TCOPR%)

| Code | Symbol | Meaning |
|------|--------|--|
| 1 | .TCSUD | Send urgent data AC3 contains pointer to table: |
| | | Word Meaning |
| | | 0 Pointer to data |
| | | 1 Count of bytes or 0 |
| | | 2 Byte to terminate output on |
| 2 | .TCPSH | Send all local buffered data immediately and set the TCP PUSH flag for the last message of the data being sent |
| 3 | .TCSPA | Set passive/active flag. AC3: Set 1 B(TC%APF) to indicate active; 0 to indicate passive |
| 4 | .TCSPP | Set persistence parameters. AC3 contains time to wait for connections. |
| | | AC3: 0 do not timeout connection |
| | | 0,,n attempt to connect for n seconds |
| | | m,,n attempt to connect for n seconds at m intervals |
| 5 | .TCSTP | Set timeout parameters. AC3 contains time to wait before a timeout and must be in range 0 to 2**18-1. If 0, no timeout will occur. |
| 7 | .TCSTS | Set type-of-service. AC3 contains the type of service desired and must be in range 0 to 2**18 - 1. Only low-order 8 bits used. |
| 10 | .TCSSC | Set security and compartment levels. AC3 contains the security level (16 bits, right-justified) in the left half and the compartment level (16 bits, right-justified) in the right half. |
| 12 | .TCSPC | Set PSI channels. AC3 contains 4 6-bit channel assignments; specify 77 octal to disable interrupt on given channel. |
| | | Flag Meaning |
| | | TC%TPU Urgent data channel (1st byte) |
| | | TC%TER Error channel (2nd byte) |
| | | TC%TSC State change channel (3rd byte) |
| | | TC%TXX Unused, must be 77 octal (4th byte) |

TOPS-20 MONITOR CALLS
(TCOPR%)

13 .TCRTW Read a single entry from the TCB. AC3 contains the word of the TCB that is desired. On return, AC3 contains the value of the word that was read.

TCOPR% ERROR MNEMONICS:

TCPX22: Invalid TCOPR function requested
TCPX26: Illegal Persist parameters
TCPX27: Illegal TCOPR Function on an OPEN TCP JFN
TCPX34: TCOPR Argument
TCPX36: Illegal TCOPR Function on an UNOPEN TCP JFN
TCPX40: TCOPR Function not yet implemented
TCPX41: TCOPR DEC interrupt channels not off
TCPX42: TCOPR Invalid TCB offset
TCPX43: TCOPR Invalid argument block

Reads input from a terminal or a file into a string in the caller's address space. Input is read until either a specified break character is encountered or the byte count is exhausted, whichever occurs first.

When used for terminal input, the TEXTI call handles the following editing functions:

1. Delete the last character input (DELETE).
2. Delete back to the last punctuation character (CTRL/W).
3. Delete back to the beginning of the current line or, if the current line is empty, back to the beginning of the previous line (CTRL/U).
4. Retype the current line from its beginning or, if current line is empty, retype the previous line (CTRL/R).
5. Accept the next character without regard to its usual meaning (CTRL/V).

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Failure, error code in AC1
+2: Success, updated pointer in word .RDDBP, appropriate bits set in the left half of word .RDFLG, and updated count in word .RDDBC of the argument block

TOPS-20 MONITOR CALLS
(TEXTI)

The format of the argument block is as follows:

| Word | Symbol | Meaning |
|------|--------|---|
| 0 | .RDCWB | Count of words following this word in the argument block. |
| 1 | .RDFLG | Flag bits. (See below.) |
| 2 | .RDIOJ | Byte pointer to string, or input JFN in the left half and output JFN in the right half (if RD%JFN is on in the flag word .RDFLG). The input JFN is where the input is being read from, and the output JFN is where any output generated from character editing is placed. |
| 3 | .RDDBP | Byte pointer to string in caller's address space where input is to be placed (destination string pointer). |
| 4 | .RDDBC | Number of bytes available in the destination string (field width). |
| 5 | .RDBFP | Byte pointer to the beginning of the destination buffer. This pointer indicates the maximum limit to which the user can edit back into the buffer with DELETE, CTRL/W, or CTRL/U. This buffer is not separate (that is, is not disjoint) from the destination string. On the first TEXTI, this pointer is normally the same as the destination byte pointer (.RDDBP), but does not have to be the same. If the count in word .RDCWB is 4, then the byte pointer in word .RDDBP will be used as the pointer to the destination buffer. |
| 6 | .RDRTY | Byte pointer to the beginning of the prompting-text (CTRL/R buffer). This text, along with any text in the destination buffer, is output if the user types CTRL/R on his first line of input. If there is no CTRL/R text or the user types CTRL/R on other than the first line of input, only the text in the destination buffer will be output. The CTRL/R buffer is useful for retyping characters that preceded the user's input, such as a prompt from the program. The text in this buffer cannot be edited by the user, and if the user deletes back to the end of this buffer, his action is treated as if he has deleted all of his input. This buffer is logically |

TOPS-20 MONITOR CALLS
(TEXTI)

adjacent to the destination buffer, but may be physically disjoint from it. When the CTRL/R buffer is disjoint, it must be terminated with a null byte.

- 7 .RDBRK Address of a 4-word block of break character mask bits. If a bit is on in the mask, then the corresponding character is considered a break character. Any bits set in this mask override break characters set in the flag word.

The mask occupies the leftmost 32 bits of each word, thereby allowing a mask of 128 bits. The rightmost 4 bits of each word are ignored. The mapping is from left to right. The ASCII character set maps into this 128-bit mask.

If this word is zero, there is no break character set mask defined.

- 10 .RDBKL Byte pointer to the backup limit in the destination buffer. This pointer indicates the position in the destination buffer to which the user can edit back without being informed. This pointer is used to indicate to the program that previously parsed text has been edited and may need to be reparsed by the program. The pointer can either be equal to the start of the buffer pointer (.RDBFP) or to the destination string pointer (.RDDBP) or be between these two pointers.

Words 5 through 10 (.RDBFP through .RDBKL) in the argument block are optional. A zero in any of the words means that no pointer has been given.

The illustration below is a logical arrangement of the CTRL/R and destination buffers, with the placement of the pointers when they are given as not being equal. Remember that the CTRL/R buffer does not have to be adjacent to the destination buffer and that two or more of these pointers can be equal.

TOPS-20 MONITOR CALLS
(TEXTI)

| | | |
|----|--------|---|
| 6 | RD%JFN | JFNs have been given for the source designator (word .RDIOJ of the argument block). If this bit is not set, the source designator is a pointer to a string. |
| 7 | RD%RIE | Return to user program if the input buffer is empty. If this bit is not set, the TEXTI call waits for more input. |
| 8 | RD%BBG | Not used |
| 9 | RD%BEG | Causes TEXTI to return when the .RDBKL pointer is reached and TEXTI is about to wait for more input. |
| 10 | RD%RAI | Convert lowercase input to uppercase input. |
| 11 | RD%SUI | Suppress the CTRL/U indication if user types a CTRL/U (that is, do not print XXX and on display terminals, do not delete the characters from the screen). |
| 15 | RD%NED | Suppress the editing functions of editing characters (for example, CTRL-R, CTRL-U) that are in the user-supplied break mask. |

On a successful return, the following bits can be set in word 1 (.RDFLG) of the argument block:

| Bit | Symbol | Meaning |
|-----|--------|---|
| 12 | RD%BTM | A break character terminated the input. If this bit is not set, the input was terminated because the byte count was exhausted. |
| 13 | RD%BFE | Control was returned to the user program because the user tried to delete beyond the beginning of the destination buffer and RD%RND was on in the call. |
| 14 | RD%BLR | The backup limit for editing was reached. |

TEXTI ERROR MNEMONICS:

ARGX17: Invalid argument block length
 RDTX1: Invalid string pointer
 IOX11: Quota exceeded
 IOX34: Disk full
 IOX35: Unable to allocate disk - structure damaged

TOPS-20 MONITOR CALLS
(TFORK)

Sets and removes monitor call intercepts (JSYS traps) for the given inferior processes.

When the process attempts to execute a call on which an intercept has been set, that process is suspended before it executes the call. Once the process is suspended, the monitor passes control to the closest superior process that is monitoring the execution of that call.

The superior process can then use the RTFRK call to determine which process caused the interrupt, and how to handle the interrupt. It can use any of the process manipulation calls, and then use the UTFRK call to resume the suspended inferior process.

Alternatively, the superior can simply decide to resume the inferior and allow it to execute the call. In this case, the next higher superior process monitoring the intercepted call receives an interrupt, and control is passed to that superior. If each superior process monitoring the call decides to resume the suspended process without changing its PC word, then the suspended process is allowed to execute the monitor call as it normally would.

Note that an RTFRK should be performed when an interrupt is received, or the monitored process will not trap again.

RESTRICTIONS: Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled for use on execute-only processes.

ACCEPTS IN AC1: Function code in the left half, and process handle in the right half

AC2: Software interrupt channel number in the left half, and size (in bits) of the monitor call bit table

AC3: Address of monitor call bit table

RETURN +1: Always

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .TFSET | Set monitor call intercepts for the given process. The calls that will be intercepted are indicated in the monitor call bit table. The given process must be frozen. This function is illegal for an execute-only process. |
| 1 | .TFRAL | Remove all monitor call intercepts for the given process. The process must be frozen. This function is illegal for an execute-only process. |

TOPS-20 MONITOR CALLS
(TFORK)

- | | | |
|----|--------|--|
| 2 | .TFRTP | Remove for the given process only the monitor call intercepts that are indicated in the monitor call bit table. The given process must be frozen. This function is illegal for an execute-only process. |
| 3 | .TFSPS | Set the given software channel as the channel on which to generate the interrupt. |
| 4 | .TFRPS | Return in the left half of AC2 the software channel on which the interrupt will be generated. |
| 5 | .TFTST | Test if the caller is to be intercepted when it attempts to execute monitor calls. On successful return AC2 contains -1 if it is to be intercepted or 0 if it is not to be intercepted. |
| 6 | .TFRES | Remove intercepts set for all inferiors and clear the software channel assigned to the interrupt for monitor call intercepts. |
| 7 | .TFUOO | Set monitor call intercepts for TOPS-10 monitor calls (UOOs) for the given process. The process must be frozen. This function is illegal for an execute-only process. |
| 10 | .TFSJU | Set monitor call intercepts for both the calls indicated in the monitor call bit table and the TOPS-10 monitor calls. This function is a combination of functions .TFSET and .TFUOO. The given process must be frozen. This function is illegal for an execute-only process. |
| 11 | .TFRUU | Remove monitor call intercepts for the TOPS-10 monitor calls. The given process must be frozen. |

To set monitor call intercepts, the process must first issue .TFSPS (code 3). Then, .TFSET (code 0), .TFUOO (code 7) or .TFSJU (code 10) may be issued to set intercepts.

The process handle in the right half of AC1 must refer to an inferior process or must be -4 to refer to all inferiors. When intercepts are set for a given process, they also apply to all processes inferior to the given process. When a process is created, it is subject to the same intercepts as the process that created it.

If the software channel is given as 77, any intercepts bypass the given process without causing either an interrupt to its superior or a suspended state of the process.

TOPS-20 MONITOR CALLS
(TFORK)

The monitor call bit table contains a bit for each of the TOPS-20 monitor calls. When a bit in the table is on, the corresponding monitor call is to be intercepted when the given process attempts to execute it. If the bit is off, the corresponding monitor call will not be intercepted. The size of the bit table is 1000(octal) bits.

A process can remove only the intercepts it previously set; it cannot remove intercepts that other processes set.

When the process being monitored attempts to execute the trapped-for JSYS, the process and its inferiors enter a suspended state. This suspended state differs from the normal "frozen" state of a process in the following ways:

1. The inferiors of the monitored process are not frozen and continue to operate.
2. The monitored process is resumed with the UTFRK monitor call. RFRK will not resume the process.
3. All interrupts for the monitored process are queued and are acted upon immediately after the UTFRK monitor call.

After the suspension of the monitored process, the superior process may do one of the following:

1. Allow the monitored process to resume execution of the intercepted JSYS.
2. Make changes in the working environment of the monitored process and allow that process to resume execution of the intercepted JSYS.
3. Execute the intercepted JSYS on behalf of the monitored process, and then allow the monitored process to continue.

The user interface to the monitor call intercept facility is provided for by three JSYSs:

1. TFORK (trap)
2. RFRK (read)
3. UTFRK (untrap)

Generates an illegal instruction interrupt on error conditions below.

TFORK ERROR MNEMONICS:

FRKH8: Illegal to manipulate an execute-only process
TFRKX1: Invalid function code

TOPS-20 MONITOR CALLS
(TFORK)

TFRKX2: Unassigned process handle or not immediate inferior
TFRKX3: Process not frozen

Blocks the current process for the specified elapsed time or until awakened by a TWAKE monitor call, whichever occurs first.

ACCEPTS IN AC1: 0 in the left half, and maximum number of seconds to block in the right half

RETURNS +1: Never
 +2: Always, with time expired or TWAKE call occurred

Returns the amount of time since the system was last restarted.

RETURNS +1: Always, with time (in milliseconds) right-justified in AC1, and divisor to convert the time to seconds in AC2. AC2 always contains 1000; thus, it is not necessary to examine its contents.

This is a monotonically increasing number (when the system is running) independent of any resets of the time and date.

Controls the amount of time either a process within a job or the entire job can run. An interrupt is generated when the time has elapsed.

TOPS-20 MONITOR CALLS
(TIMER)

Only one process in the job is allowed to time the entire job. If the job is already being timed, an error is given if another process attempts to time the job. An error is also given if a process other than the one that set the runtime limit of the job attempts to remove that limit.

ACCEPTS IN AC1: Process handle in the left half, and function code in the right half.

AC2: Time at which to generate an interrupt. See the individual function descriptions for the specific arguments.

AC3: Number of the software channel on which to generate an interrupt when the time has expired.

RETURNS +1: Failure, error code in AC1

+2: Success

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .TIMRT | Specify the total runtime of the entire job. This function allows one process within a job to time the entire job. AC2 contains the total runtime in milliseconds that the job can accumulate before an interrupt is generated on the specified channel. If AC2 contains 0, the limit on the runtime of the job is removed. The process handle given in AC1 must be .FHJOB (-5). |
| 1 | .TIMEL | Specify an elapsed time after which an interrupt is generated for the given process. AC2 contains the number of milliseconds that can now elapse before the interrupt is generated on the specified channel. |
| 2 | .TIMDT | Specify an exact time at which an interrupt is generated for the given process. AC2 contains the internal format (see section 2.6.3) of the date and time when the interrupt is to be generated. |
| 3 | .TIMDD | Remove any pending interrupt requests that are to occur for the process at the given time. AC2 contains the internal format (see section 2.9.2) of the date and time of the interrupt request to be removed. AC3 is not used for this function. |

TOPS-20 MONITOR CALLS
(TIMER)

- 4 .TIMBF Remove any pending interrupt requests that are to occur for the process before the given time. AC2 contains the internal format (see section 2.9.2) of the date and time. AC3 is not used for this function.
- 5 .TIMAL Remove all pending requests for the given process including the runtime limit on the entire job. AC3 is not used for this function.

The runtime limit for a job can be obtained via the GETJI monitor call (contents of word .JIRT on return). If the job's time limit has been exceeded, the value returned by the GETJI call will be zero.

TIMER ERROR MNEMONICS:

TIMX1: Invalid function
TIMX2: Invalid process handle
TIMX3: Time limit already set
TIMX4: Illegal to clear time limit
TIMX5: Invalid software interrupt channel number
TIMX6: Time has already passed
TIMX7: No space available for a clock
TIMX8: User clock allocation exceeded
TIMX9: No such clock entry found
TIMX10: No system date and time

Controls terminal linking. (See Section 2.4.9.5 for more information.)

RESTRICTIONS: Some functions require WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: B0(TL%CRO) Clear link from remote to object designator. If the remote designator is -1, all remote links to the object designator are cleared.

 B1(TL%COR) Clear link from object to remote designator. If the remote designator is -1, links from the object to all remote designators are cleared.

TOPS-20 MONITOR CALLS
(TLINK)

B2(TL%EOR) Establish link from object to remote designator.

B3(TL%ERO) Establish link from remote to object designator.

B4(TL%SAB) Examine B5(TL%ABS) to determine the setting of the object designator's accept link bit. If this bit is off, B5 is ignored.

B5(TL%ABS) Set the object designator's accept link bit. When B4(TL%SAB) is on, the object designator is accepting links; if TL%ABS is off the object designator is refusing links.

B6(TL%STA) Examine B7(TL%AAD) to determine the setting of the object designator's accept advice bit. If this bit is off, B7 is ignored.

B7(TL%AAD) Set the object designator's accept advice bit. When B6(TL%STA) is on, the object designator is accepting advice if TL%AAD is on and refusing advice if TL%ADD is off.

B18-B35 Object designator
(TL%OBJ)

AC2: Remote designator in the right half

RETURNS +1: Failure, error code in AC1

+2: Success

The object and remote designators must be either 4xxxxx or -1. An object designator of -1 indicates the controlling terminal. The following restrictions apply if the process does not have WHEEL capability enabled:

1. The object designator must specify this terminal.
2. The object-to-remote link must be specified before or at the same time as the remote-to-object link.

If the accept bit of the remote designator is not set, a link from the object-to-remote designator causes the remote designator's bell to ring. If the remote designator does not set the accept bit within 15 seconds, the TLINK call returns an error.

TOPS-20 MONITOR CALLS
(TLINK)

When terminals are linked together and a character is typed on one terminal, the same ASCII character code is sent to all terminals in the link. The character always appears in the output buffers of all terminals regardless of the current mode of each individual terminal. The character is sent according to the data mode and terminal type of the terminal that originates the character. For example, if one terminal originates a TAB and has mechanical tabs set, all terminals in the link receive the ASCII code for a TAB in their output buffers.

TLINK ERROR MNEMONICS:

DESX1: Invalid source/destination designator
TLNKX1: Illegal to set remote to object before object to remote
TLNKX2: Link was not received within 15 seconds
TLNKX3: Links full
TTYX01: Line is not active

Returns various flags and parameters in the monitor's data base. In most cases, flag-oriented items return a 1 in AC2 if the flag is set and a 0 in AC2 if the flag is cleared. In a few cases (noted in the text), flag-oriented items return the appropriate bit set or cleared in AC2. Value-oriented items return the value of the parameter in AC2.

ACCEPTS IN AC1: Function code

RETURNS +1: Always, with value of the function in AC2

The codes for the functions are as follows:

| Code | Symbol | Meaning |
|------|--------|---|
| 0 | .SFFAC | FACT file entries are allowed. |
| 1 | .SFCDE | CHECKD found errors. |
| 2 | .SFCDR | CHECKD is running. |
| 3 | .SFMST | Manual start is in progress. |
| 4 | .SFRMT | Remote LOGINS (dataset lines) are allowed. |
| 5 | .SFPTY | PTY LOGINS are allowed. |
| 6 | .SFCTY | CTY LOGINS are allowed. |
| 7 | .SFOPR | Operator is in attendance. |
| 10 | .SFLCL | Local LOGINS (hardwired lines) are allowed. |
| 11 | .SFBTE | Bit table errors found on startup. |
| 12 | .SFCRD | Users can change nonprivileged directory |

TOPS-20 MONITOR CALLS
(TMON)

parameters with the CRDIR monitor call.

| | | |
|----|--------|---|
| 13 | .SFNVT | TCP/IP terminal LOGINS are allowed. |
| 14 | .SFWCT | WHEEL LOGINS on CTY are allowed. |
| 15 | .SFWLC | WHEEL LOGINS on local terminals are allowed. |
| 16 | .SFWRM | WHEEL LOGINS on remote terminals are allowed. |
| 17 | .SFWPT | WHEEL LOGINS on PTYs are allowed. |
| 20 | .SFWNV | WHEEL LOGINS on network virtual terminals (NVT) are allowed. |
| 21 | .SFUSG | USAGE file entries are allowed. |
| 22 | .SFFLO | Disk latency optimization using the RH20 backup register is enabled. This feature is not to be enabled unless the M8555 board of the RH20 is at Revision Level D AND either of the KL10-C processor is at Revision Level 10 or KL10-E processor is at Revision Level 2. |
| 23 | .SFMTA | MOUNTR magtape allocation is enabled. |
| 24 | .SFMS0 | System message level 0 is set. |
| 25 | .SFMS1 | System message level 1 is set. |
| 26 | .SFBGS | Operator messages are sent to CTY; if off, such messages as BUGINF, BUGCHK, and "resource low" are sent to OPR terminals, rather than the CTY. |
| 27 | .SFMCB | DECnet logins allowed |
| 30 | .SFDPR | Disk preallocation is enabled. |
| 31 | .SFLAT | LAT LOGINS are allowed. |
| 32 | .SFWLT | WHEEL LOGINS on LAT terminals are allowed. |
| 44 | .SFNTN | TCP/IP is on. |
| 45 | .SFNDU | TCP/IP will be reinitialized if it is down. |
| 46 | .SFNHI | TCP/IP host table will be initialized. |
| 47 | .SFTMZ | Local time zone |
| 50 | .SFLHN | TCP/IP local host number |
| 51 | .SFAVR | Account validation is running on this system. |
| 52 | .SFSTS | Status reporting is enabled. |
| 53 | .SFSOK | GETOK% defaults |

Required in AC2: GETOK% function code

Returned in AC2: Flags,,GETOK% function code

Flags:

| Bit | Symbol | Meaning |
|-----|--------|---|
| B0 | SF%EOK | 0 = Access checking is disabled 1 = Access checking is enabled |
| B1 | SF%DOK | 0 = Access is denied if checking disabled 1 = Access is allowed if checking disabled |

TOPS-20 MONITOR CALLS
(TMON)

Installation-defined function codes (400000+n) must be enabled/disabled by using function code 400000, regardless of the installation-defined function code given in the GETOK% call. See the description of the GETOK% JSYS for GETOK% function codes.

| | | |
|----|--------|---|
| 54 | .SFMCY | Maximum offline expiration period in days in days for ordinary files (tape recycle period). |
| 55 | .SFRDU | Read date update function data |
| 56 | .SFACY | Maximum offline expiration period in days for archive files (tape recycle period). |
| 57 | .SFRTW | File-retrieval requests that are waiting for the retrieval should fail rather than wait. |
| 60 | .SFTDF | Tape mount controls |

Flags:

| Bit | Symbol | Meaning |
|-----|--------|--|
| B0 | MT%UUT | 1 = unload unrecognizable tapes 0 = treat unrecognizable tapes as unlabeled |

| | | |
|----|--------|--|
| 61 | .SFWSP | Enable working set preloading |
| 62 | .SFDST | Daylight Saving Time conversion method |

| Value | Symbol | Meaning |
|-------|--------|----------------------------------|
| 0 | .DSTAU | Perform automatic DST changeover |
| 1 | .DSTNV | Never perform DST changeover |
| 2 | .DSTAL | Always perform DST conversion |

| | | |
|----|---------|---|
| 63 | | Reserved for DIGITAL. |
| 64 | .SFMSD | MSCP access for disk drive; see the SMON% monitor call for a description of the argument block. Upon return, AC2 contains 1 if the drive is ALLOWED; 0 if RESTRICTED. |
| 65 | .SFSPR | Read SPEAR event counter |
| 66 | .SFCOT | Read time between carrier off event (including network connection being broken) and automatic logout of the job. AC2 is the time in milliseconds. |
| 67 | .SFHU0 | Hang up action for jobs not logged in AC2: 0 to not hang up; 1 to hang up |
| 70 | .SFHU1 | Hang up action for jobs logged in AC2: 0 to not hang up; 1 to hang up |
| 71 | .SFXEC | Flag word for configurations for the EXEC (see SMON) |
| 72 | .SFSEA | Read Ethernet address (see SMON) |
| 73 | .SFD CD | Read "don't care disk" status (see SMON) |
| 74 | .SFLTS | Read Local Area Transport (LAT) state (see SMON) |
| 75 | .SFCLU | Read "on/off" status of remote INFO%. |

TOPS-20 MONITOR CALLS
(TMON)

| | | |
|-----|--------|--|
| 76 | .SFMTG | Read "on/off" status of remote TTMSG%. |
| 77 | .SFOFS | Read the offline structure timeout interval in seconds; 0 implies disabled. |
| 100 | .SFLGS | The login structure feature is enabled. |
| 101 | .SFMPL | Read minimum password length. Minimum length must be 1 to 39 characters. AC2: Minimum length or 0 to disable. |
| 102 | .SFACJ | WHEEL or OPERATOR capability required to read the setting of this function. AC 2: 0 - ACJ is running, in monitor context AC 2: 1 - ACJ is running, not in monitor context AC 2: -1 - ACJ is not running |
| 103 | .SFPEX | Reads password expiration setting. See corresponding SMON% function. |
| 104 | .SFPWD | Reads dictionary enable/disable setting. See corresponding SMON% function. |
| 105 | .SFHDT | Reads the state of hangup on DETACH. See the corresponding SMON% function. |

The SMON monitor call can be used to set various monitor flags.

Generates an illegal instruction interrupt on error conditions below.

TMON ERROR MNEMONICS:

TMONX1: Invalid TMON function

Sends a message to a specified terminal on a specified system or to all terminals on all systems.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled to send to all terminals. Messages sent by privileged callers may contain a maximum of 581 characters; messages sent by non-privileged callers may contain a maximum of 526 characters.

ACCEPTS IN AC1: .TTDES + local TTY number or -1 to send to all local terminals or

B1(TT%REM) Indicates a remote send.

B13-B17(.TTCIN) Indicates the CI node number. Use

TOPS-20 MONITOR CALLS
(TTMSG)

.CSALL (37,,0) for all nodes.

B18-B35(.TTTTY) .TTDES +TTY number or 777777 for
all terminals on specified node(s).

AC2: Byte pointer to string to be sent

RETURNS +1: Always

The message being sent is not formatted to the current width setting of the destination terminal.

The TTMSG monitor call is a no-op if the specified terminal does not exist.

Generates an illegal instruction interrupt on error conditions below.

TTMSG ERROR MNEMONICS:

GTDIX1: WHEEL or OPERATOR capability required
TTMSX1: Could not send message within timeout interval
TTMSX2: User is refusing messages and/or links
TTMSX3: Invalid CI node number
TTMSX4: Remote node not accepting remote sendalls

Wakes the specified job that is blocked because of the execution of a THIBR call. If more than one process in a job is blocked because of a THIBR call, execution of the TWAKE call causes any one of the processes to be awakened.

ACCEPTS IN AC1: 0 in the left half, and number of job to be awakened in the right half

RETURNS +1: Failure, error code in AC1
+2: Success, signal sent. Job will be awakened immediately if blocked by a THIBR call or as soon as next THIBR call is executed.

TWAKE ERROR MNEMONICS:

ATACX1: Invalid job number

TOPS-20 MONITOR CALLS
(UFPGS)

Updates pages of the specified file. This monitor call is used to guarantee that a certain sequence of file pages has been written to the disk before any other operation is performed.

ACCEPTS IN AC1: JFN in the left half, and file page number of the first page to be updated in the right half

AC2: Flags,,count of number of sequential pages to update

RETURNS +1: Failure, error code in AC1

+2: Success, all modified pages are written to disk. Words .FBADR and .FBCTL of the FDB are updated, if necessary.

Flags:

| Bit | Symbol | Meaning |
|-----|--------|---|
| 0 | UF%NOW | Allows performing a UFPGS call without blocking. The JSYS will not block even if some pages need to be written to disk. |
| 1 | UF%FSH | Flush the incore copy of pages. |

If UF%NOW is not set, the UFPGS call causes the process to block until all writes to the disk are completed.

UFPGS ERROR MNEMONICS:

UFPGX1: File is not opened for write
DESX3: JFN is not assigned
DESX4: Invalid use of terminal designator or string pointer
DESX7: Illegal use of parse-only JFN or output wildcard-designators
DESX8: File is not on disk
LNGFX1: Page table does not exist and file not open for write
IOX11: Quota exceeded
IOX34: Disk full
IOX35: Unable to allocate disk - structure damaged

Controls accounting on the system by writing entries into the system's

TOPS-20 MONITOR CALLS
(USAGE)

data file. All entries to the data file are made with this call. Examples of the types of entries entered into the data file are disk storage usage for regulated structures, input and output spooler usage, job session entry, and date and time changes.

The file written by the USAGE call is an intermediate binary file, which is converted by a system program to the final ASCII file. Each entry in the final file is at least two records long, each record being defined as a string of ASCII characters terminated with a line-feed character. The first record contains system and file information; its format is the same for all entries. Subsequent records contain data pertaining to the entry; their formats vary according to the particular data being entered.

See the USAGE File Specification for additional information on the system's data file.

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

ACCEPTS IN AC1: Function code

AC2: Function argument or address of record descriptor block

RETURNS +1: Always

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .USENT | Write an entry into the system's data file. AC2 contains the address of the record descriptor block. |
| 1 | .USCLS | Close the system's data file, which is named PS:<ACCOUNTS>SYSTEM-DATA.BIN. No additional entries are recorded into this file, and a new SYSTEM-DATA.BIN is opened for subsequent entries. |
| 2 | .USCKP | Perform a checkpoint of all jobs. Data recorded during a checkpoint includes the billable data (connect time and runtime, for example) accumulated during the job session. The session starts from time of login or the last SET ACCOUNT command, and ends at the time this function is performed. The data collected on a LOGIN or SET ACCOUNT command is entered into the session entry in the data file. The default checkpoint interval is 10 minutes. |
| 3 | .USLGI | Initialize a checkpoint entry for the job. This |

TOPS-20 MONITOR CALLS
(USAGE)

function is used internally by the LOGIN monitor call. AC2 contains the address of the record descriptor block.

- 4 .USLGO Terminate the checkpoint entry for the job and write an entry into the system's data file, which is named PS:<ACCOUNTS>SYSTEM-DATA.BIN. This function is used internally by the LGOUT monitor call. AC2 contains the address of the record descriptor block.
- 5 .USSEN Terminate the current session, write an entry into the system's data file, which is named PS:<ACCOUNTS>SYSTEM-DATA.BIN, and initialize a new checkpoint entry for the job. This function is used internally by the CACCT monitor call. AC2 contains the address of the record descriptor block.
- 6 .USCKI Set the checkpoint time interval. AC2 contains the interval in minutes.
- 7 .USENA Install the accounting data base from the file named PS:<SYSTEM>ACCOUNTS-TABLE.BIN into the running monitor. The ACTGEN program uses this file to generate the list of valid accounts.
- 10 .USCAS Change accounting shift. This function will perform a "session end" function for every active job.
- 11 .USSAS Set accounting shifts. Sets the times when automatic accounting shift changes are to occur. This function takes an argument in AC2 which is a pointer to a block of the following format:

table header

table entry

...

table entry

The table header word contains the number of actual entries in the table in the left halfword, and the maximum number of table entries in the right halfword. Each table entry is one word in the following format:

B0-B6 US%DOW Days of the week that this

TOPS-20 MONITOR CALLS
(USAGE)

| | | |
|---------|--------|--|
| | | entry is in effect. Bit n is set if this entry is in effect for day n (0 = Monday). |
| B7-B17 | | Unused, must be zero. |
| B18-B35 | US%SSM | Time of day that automatic shift change should occur. Time is specified in seconds since midnight. |

The maximum number of table entries is 100 decimal.

12 .USRAS Read accounting shifts. This function returns the times of the automatic shift changes that were set with .USSAS. AC2 contains the address of an argument block that is filled in by this function. The block has the same format as the .USSAS block. Note that the right halfword (maximum size) of the table header must be specified by the user for .USRAS.

The record descriptor block, whose address is given in AC2, is set up by the UITEM. macro defined in ACTSYM.MAC. The names of all data entries are generated by this macro. The USENT. macro is used to generate the header of the record descriptor block.

The format of the data generated by the USAGE call is a list of items describing the entries in a single record. This list has a header word containing the version numbers and the type of entry. The data words follow this header with two words per data item. The list is terminated with a zero word.

Generates an illegal instruction interrupt on error conditions below.

USAGE ERROR MNEMONICS:

CAPX1: WHEEL or OPERATOR capability required
ARGX02: Invalid function
ARGX04: Argument block too small
ARGX05: Argument block too long
USGX01: Invalid USAGE entry type code
USGX02: Item not found in argument list
USGX03: Default item not allowed
USGX04: Invalid terminal line number

TOPS-20 MONITOR CALLS
(USRIO)

Places the user program into user I/O mode in order that it can execute various hardware I/O instructions. The user IOT flag is turned on in the PC of the running process. The program can leave user I/O mode by executing a JRSTF with a PC in which bit 6 is zero (for example, JRSTF @[.+1]).

RESTRICTIONS: Requires WHEEL or OPERATOR capability enabled.

RETURNS +1: Failure, error code in AC1
 +2: Success, user IOT flag is set

USRIO ERROR MNEMONICS:

CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required

Provides a method for determining if every instruction in a section of monitor code actually gets executed. This monitor call does not test the code by executing it; it confirms that a test of the code is complete by reporting the instructions that were executed during the test.

RESTRICTIONS: Requires WHEEL capability enabled.

ACCEPTS IN AC1: Function code in the left half, and length of the argument block in the right half.

 AC2: Address of the argument block

RETURNS +1: Always

The available functions are as follows:

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .UTSET | Start testing of the code. |
| 1 | .UTCLR | Stop testing of the code and update the bit map in the argument block. |

The format of the argument block is as follows:

TOPS-20 MONITOR CALLS
(UTEST)

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .UTADR | Address of the beginning of the section of code that is to be tested. |
| 1 | .UTLEN | Length of section of code that is to be tested. |
| 2 | .UTMAP | Start of bit map representing the instructions that are to be tested in the section of code. This map contains one bit for each location in the section. If a bit is on in the map, the corresponding instruction is to be tested. If a bit is off, the corresponding instruction is not to be tested. |

Locations that contain data and that would cause the section of code to execute improperly if that data were changed should not be tested.

Internally, a copy of the code being tested is placed in a buffer, which is dynamically locked down during execution of the UTEST call. The system allows any monitor routine to be tested as long as a pushdown stack to which AC P (AC17) points is set up whenever the routine is called.

After execution of the .UTCLR function, the bit map is changed to reflect the instructions that were actually executed during the test. If a bit is on in the map, the corresponding instruction was executed. If a bit is off, the corresponding instruction was not executed.

Generates an illegal instruction interrupt on error conditions below.

UTEST ERROR MNEMONICS:

CAPX3: WHEEL capability required
UTSTX1: Invalid function code
UTSTX2: Area of code too large to test
UTSTX3: UTEST facility in use by another process

Resumes the execution of a process that is suspended because of a monitor call intercept. The instruction where the execution resumes depends on the current PC word of the suspended process. To prevent the suspended process from executing the call, the superior process

TOPS-20 MONITOR CALLS
(UTFRK)

handling the intercept can change the PC word (via the SFORK or SFRKV call). Then on execution of the UTFRK call, the suspended process continues at the new PC. If the superior process handling the intercept does not change the PC word of the suspended process, then the next superior process intercepting that particular monitor call will receive the interrupt.

See the description of the TFORK JSYS for more information on the monitor call intercept facility.

ACCEPTS IN AC1: Flag bits in the left half, and process handle in the right half

RETURNS +1: Always

The flag bit that can be given in AC1 is as follows:

| Bit | Symbol | Meaning |
|-----|--------|---|
| 0 | UT%TRP | Cause a failure return for the suspended process. This return will be either the generation of an illegal instruction interrupt or the processing of an ERJMP or ERCAL instruction. |

The UTFRK monitor call is a no-op if

1. The process handle given is valid but the process specified is not suspended because of a monitor call intercept.
2. The caller is not one of the processes monitoring the suspended process and therefore is not permitted to resume the process.

Generates an illegal instruction interrupt on error conditions below.

UTFRK ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate a superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process

TOPS-20 MONITOR CALLS
(VACCT)

Verifies accounts by validating the supplied account for the given user.

RESTRICTIONS: Requires WHEEL or OPERATOR capability, unless caller is validating his current account.

ACCEPTS IN AC1: 36-bit user number, 36-bit directory number, or -1 to validate the account for the current user

AC2: Byte pointer to account string

RETURNS +1: Always, with updated pointer in AC2

Generates an illegal instruction interrupt on error conditions below.

VACCT ERROR MNEMONICS:

VACCX0: Invalid account
VACCX1: Account string exceeds 39 characters
VACCX2: Account has expired
MONX02: Insufficient system resources (JSB full)
DELFX6: Internal format of directory is incorrect
DIRX1: Invalid directory number
DIRX3: Internal format of directory is incorrect
STRX01: Structure is not mounted
OPNX9: Invalid simultaneous access
OPNX16: File has bad index block

Dismisses the current process indefinitely and does not return. If the software interrupt system is enabled for this process, the process can be interrupted out of the wait state. Upon execution of a DEBRK call, the process continues to wait until the next interrupt unless the interrupt routine changes the PC word. In this case, the process resumes execution at the new PC location. If the interrupt routine changes the PC word, it must set the user-mode bit (bit 5) of the PC word. (See Section 2.6.7.)

TOPS-20 MONITOR CALLS
(WFORK)

Causes the current process to wait for a specific inferior process or all inferior processes to terminate (voluntarily or involuntarily). A process is considered terminated if its state is either .RFHLT or .RFFPT (see RFSTS JSYS for a description of process status).

ACCEPTS IN AC1: Inferior process handle, or -4 (.FHINF) in the right half to wait for all of the inferior processes to terminate

RETURNS +1: Always, when the specified process(es) terminates

This call returns immediately if the specified process(es) has already terminated.

Generates an illegal instruction interrupt on error conditions below.

WFORK ERROR MNEMONICS:

FRKHX1: Invalid process handle

FRKHX2: Illegal to manipulate a superior process

Compares a possibly wild string (one containing wild-card characters) against a non-wild string to see if the latter matches the wild string. For example, "AND" would be a legal match for the wild string "A*D". Likewise "AND" would be a legal match for the wild string "A%%". The WILD% JSYS will also compare a possibly wild file specification with a non-wild file specification. (See Section 2.2.3 for a description of wild-card characters.)

ACCEPTS IN AC1: Flags in the left half, function in the right half

AC2: Wild argument - JFN or byte pointer to string

AC3: Non-wild argument - JFN or byte pointer to string

RETURNS +1: Always, with information returned in AC1

The available functions are as follows:

TOPS-20 MONITOR CALLS
(WILD%)

| Code | Symbol | Meaning |
|------|--------|--|
| 0 | .WLSTR | <p>Compare a non-wild string against a wild string. AC2 contains a byte pointer to a wild string and AC3 contains a byte pointer to a non-wild string. By default, the comparison is made without regard to what kind of characters the strings contain. Thus tabs, spaces, and carriage returns, for example, are treated just as letters are. The following flag can be set in AC1:</p> <p>B0(WL%LCD) Lower case characters are to be treated as distinct from upper case letters. If this bit is not set, a lower case character will match the corresponding upper case character.</p> <p>On return, AC1 contains zero if a match occurred, or the following flags if no match occurred:</p> <p>B0(WL%NOM) If set, this bit indicates that the non-wild string did not match the wild string.</p> <p>B1(WL%ABR) If set, this bit indicates that the non-wild string is not matched, but is an abbreviation of the wild string. If this bit is set, it implies that bit WL%NOM is also set.</p> |
| 1 | .WLJFN | <p>Compare a non-wild file specification against a wild file specification. AC2 contains a JFN with flags (as returned by GTJFN) for the wild file and AC3 contains a JFN (without flags) for the non-wild file. On return, AC1 contains zero if a match occurred. Otherwise, the following flags are returned (in AC1) to indicate which parts of the file specification do not match:</p> <p>B1(WL%DEV) Device field does not match B2(WL%DIR) Directory field does not match B3(WL%NAM) Name field does not match B4(WL%EXT) File type does not match B5(WL%GEN) Generation number does not match</p> |

If a parse-only JFN is given (see section 2.2.3), and one of the fields is not specified (such as a file name), that field will be treated as a null field. Thus the filenames PS:<DBELL>FOO.BAR.3 and PS:<DBELL>.BAR.3 will not match.

TOPS-20 MONITOR CALLS
(WILD%)

WILD% ERROR MNEMONICS:

DESX3: JFN is not assigned
RDTX1: Invalid string pointer
ARGX02: Invalid function
ARGX22: Invalid flags

Manages the working set of a process.

ACCEPTS IN AC1: Function code
AC2: Pointer to argument block
AC3: Process handle

RETURNS +1: Always

The available functions are:

| Code | Symbol | Meaning |
|------|--------|--|
| 1 | .WSCLR | Clear the working set of the calling process. This function is similar to the RWSET% call. |
| 2 | .WSRMV | Remove specified pages from the working set of the calling process. Usually these pages are then swapped out of memory. The argument block specifies the pages to remove. |
| 3 | .WSGET | Get pages into memory for the calling process. The process's working set is not affected. The pages specified by the argument block are brought into memory so that an immediate reference will not cause the process to be blocked. This function is identical to the PM%PLD function of the PMAP% call. This function does not create pages and thus is not valid for nonexistent pages. |
| 4 | .WSRWS | Read working set information for the calling process or one of its inferiors. The information is returned in the argument block, with the left half of the first word containing the count of the number of pairs returned. If the caller did not provide enough room for returning the working set, the count will reflect the number of pairs that would be needed. This function may change the |

TOPS-20 MONITOR CALLS
(WSMGR%)

working set for the calling process since the function returns data into the user's address space. The data returned reflects the process's working set at some time during the execution of the call.

The argument block has the following format:

| Offset | Contents |
|--------|---|
| 0 | Count of 2-word working set group descriptors |
| 1 | Count of pages in group 1 |
| 2 | First page of group 1 |
| | . |
| | . |
| | . |
| 2N-1 | Count of pages in group N |
| 2N | First page of group N |

Generates an illegal instruction interrupt on error conditions below.

WSMGR% ERROR MNEMONICS:

ARGX06: Invalid page number
ARGX24: Invalid count
FRKHX1: Invalid process handle

Gets an extended special entry vector that has been set to allow use of TOPS-10 Compatibility and RMS entry vectors in nonzero sections. (See the RMS Manual for more information on the Record Management System.)

ACCEPTS IN AC1: Vector type code,,fork handle

RETURNS +1: Always, with length of entry vector in AC2, and flags in bits 0-5 of AC3, address of entry vector in bits 6-35 of AC3.

TOPS-20 MONITOR CALLS
(XGSEV%)

Generates an illegal instruction trap on error return.

See XSSEV% for a list of vector type codes.

Flags returned in bits 0-5 of AC3 are the same as those listed for XSSEV%.

XGSEV% ERROR MNEMONICS

XSEVX1: Illegal vector type

Returns the page-fail words. This monitor call allows a program to retrieve information about a previous page-fail trap.

ACCEPTS IN AC1: Process handle

AC2: Address of block in which to return data. The first word of the data block must contain the number of words in the argument block. The other words of the data block should contain zero.

RETURNS +1: Always, with page-fail data returned in the data block

The data block has the following format:

```
!=====  
!      Length of the data block, including this word      !  
!=====  
! page-fail flags !  
!-----!  
!              Address that referenced the page              !  
!=====  
! MUUO opcode & AC !  
!-----!  
!      !      30-bit Effective address of the MUUO      !  
!=====  
!-----!
```

B0(PF%USR) page failure on a user-mode reference
B1(PF%WTF) page failure on a write reference

TOPS-20 MONITOR CALLS
(XGTPW%)

This information allows a program to determine the exact cause of a memory trap and the effective virtual address that caused the trap. This information is sufficient to enable the program to continue, if desired, when the cause of the trap has been removed.

Generates an illegal instruction interrupt on error conditions below.

GTRPW ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Returns the entry vector of the specified process. The process can be one that runs in one or more sections of memory. (See Section 2.7.3.)

ACCEPTS IN AC1: Process handle

RETURNS +1: Always, with length of the entry vector in AC2,
address of the entry vector in AC3.

The XSVEC% monitor call can be used to set the entry vector of a process that runs in one or more sections of memory.

Generates an illegal instruction interrupt on the following error conditions:

XGVEC% ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle

Performs monitor data retrieval functions, allowing the process to obtain various function-related data from the monitor. This monitor call allows access to data in extended sections of the monitor.

TOPS-20 MONITOR CALLS
(XPEEK%)

RESTRICTIONS:20 Requires WHEEL, OPERATOR, or MAINTENANCE capability enabled.

ACCEPTS IN AC1: Address of argument block

RETURNS +1: Always

The available functions are described below.

| Code | Symbol | Function |
|------|--------|--|
| 1 | .XPPEK | Transfers a block of words from the monitor's address space to the user's address space. |

The desired monitor words must exist on pages that have read access.

The argument block has the following format:

| Word | Symbol | Meaning |
|------|--------|--|
| 0 | .XPABL | Length of argument block including the header. |
| 1 | .XPFNC | Function code. |
| 2 | .XPCN1 | Count of words to transfer. Current maximum is one section. |
| 3 | .XPCN2 | Count of words actually transferred. This differs from the number requested if an error, such as an illegal write, occurs during the transfer. |
| 4 | .XPMAD | Location in the monitor's address space from which to start the transfer. |
| 5 | .XPUAD | Location in the user's address space into which to start the transfer. |

Generates an illegal instruction interrupt on error conditions below.

XPEEK% ERROR MNEMONICS:

CAPX2: WHEEL, OPERATOR, or MAINTENANCE capability required
PEEKX2: Read access failure on monitor page
ARGX04: Argument block too small

TOPS-20 MONITOR CALLS
(XRIR%)

Reads the addresses of the channel and priority level tables for the specified process. (See Section 2.6.3.) These addresses must be set with the XSIR% monitor call.

ACCEPTS IN AC1: Process handle

AC2: Address at which to begin the argument block

RETURNS +1: Always. The argument block contains the information stored in the Process Storage Block.

The format of the returned argument block is as follows:

```
!=====  
! Length of the argument block, including this word !  
!-----!  
! Address of the interrupt level table !  
!-----!  
! Address of the channel table !  
!-----!  
!=====!
```

To see the format of the channel and interrupt level tables, see Section 2.6.3.

Acquires a handle on a page in a process to determine the access allowed for that page.

ACCEPTS IN AC1: Process handle in the left half, and zero in the right half

AC2: Address of the argument block

RETURNS +1: Always, with a handle on the page in word 1 of the returned data block, and access information in word 2. The handle in word 1 is a process/file designator in the left half and a page number in the right half.

The argument block addressed by AC2 has the following format:

TOPS-20 MONITOR CALLS
(XRMAP%)

```

!=====!
!   Length of the argument block, including this word   !
!=====!
!           number of pages on which to return data           !
!-----!
!           number of the first page in this group           !
!-----!
!           address at which to return the data block           !
!=====!
\           .           \
\           :           \
\           .           \
!=====!
! number of pages in this group on which to return data !
!-----!
!           number of the first page in this group           !
!-----!
!           address at which to return the data block           !
!=====!

```

The number of words in the argument block is three times the number of groups of pages for which you want access data, plus one. Each group of pages requires three arguments: the number of pages in the group, the number of the first page in the group, and the address at which the monitor is to return the access data.

The address to which the monitor returns data should be in a section of memory that already exists.

The access information returned for each group of pages specified in the argument block is the following:

```

B2(RM%RD)      read access allowed
B3(RM%WR)      write access allowed
B4(RM%EX)      execute access allowed
B5(RM%PEX)     page exists
B9(RM%CPY)     copy-on-write access

```

XRMAP% returns a -1 for each page specified in the argument block that does not exist. It also returns a zero flag word for each such page.

Generates an illegal instruction interrupt on error conditions below.

XRMAP% ERROR MNEMONICS:

```

FRKHX1:   Invalid process handle
ARGX17:   Invalid argument block length

```


TOPS-20 MONITOR CALLS
(XSFRK%)

Starts the specified process in a nonzero section of memory. If the process is frozen, the XSFRK% call changes the PC but does not resume the process. The RFORK call must be used to resume execution of the process.

ACCEPTS IN AC1: Flags,,process handle

Flags:

SF%CON(1B0) Continue a process that has halted. If SF%CON is set, the address in AC3 is ignored and the process continues from where it was halted.

AC2: PC flags in the left half, 0 in the right half

AC3: Address to which this call is to set the PC

RETURNS +1: Always

The SFRKV monitor call can be used to start a process at a given position in its entry vector.

Generates an illegal instruction interrupt on error conditions below.

XSFRK% ERROR MNEMONICS:

FRKH1: Invalid process handle
FRKH2: Illegal to manipulate a superior process
FRKH3: Invalid use of multiple process handle
FRKH5: Process has not been started
FRKH8: Illegal to manipulate an execute-only process

Sets the addresses of the channel and priority level tables for the specified process. (See Section 2.6.3.) This process can run in one or more sections of memory.

ACCEPTS IN AC1: Process handle

AC2: Address of the argument block

TOPS-20 MONITOR CALLS
(XSIR%)

RETURNS +1: Always. The addresses in the argument block are stored in the Process Storage Block.

The format of the argument block is as follows:

```
!=====!  
! Length of the argument block, including this word !  
!-----!  
! Address of the interrupt level table !  
!-----!  
! Address of the channel table !  
!=====!
```

To see the format of the channel and interrupt level tables, see Section 2.6.3.

If the contents of the tables are changed after execution of the XSIR% call, the new contents will be used on the next interrupt.

The XRIR% monitor call can be used to obtain the table addresses set with the XSIR% monitor call.

Generates an illegal instruction interrupt on error conditions below.

XSIR% ERROR MNEMONICS:

- ARGX04: Argument block too small
- ARGX05: Argument block too long
- SIRX1: Table address is not greater than 20
- XSIRX2: Level table crosses section boundary
- FRKHX1: Invalid process handle
- FRKHX2: Illegal to manipulate a superior process
- FRKHX3: Invalid use of multiple process handle
- FRKHX8: Illegal to manipulate an execute-only process

Allows setting of extended special entry vector for use with TOPS-10 Compatibility and RMS entry vectors in nonzero sections. (See the RMS Manual for more information on the Record Management System.)

ACCEPTS IN AC1: Vector type code,,fork handle

AC2: Length of entry vector

TOPS-20 MONITOR CALLS
(XSSEV%)

AC3: Flags in bits 0-5, address of entry vector in bits 6-35

RETURNS +1: Always

In order to be called from any section, the called program must provide extended format PC and UUU words. A flag in the call specifies whether the program expects new or old format words. Old format words should only be used for old versions of the program still running in Section 0.

The vector type codes supplied in the left half of AC1 are as follows:

| Code | Symbol | Meaning |
|------|--------|-----------------------|
| 0 | .XSEVC | TOPS-10 Compatibility |
| 1 | .XSEVD | RMS |

The flags set in bits 0-5 of AC3 are:

| Flag | Symbol | Meaning |
|------|--------|---|
| B1 | XS%EEV | Extended entry vector. If this bit is on, the entry vector points to a 2-word extended PC and to an extended format UUU word. If this bit is off, the entry vector points to old format PC and UUU words. |

XSSEV% ERROR MNEMONICS:

XSEVX1: Illegal entry vector type
XSEVX2: Invalid entry vector length

Sets or clears the entry vector of the specified process. The process can be one that runs in one or more sections of memory. (See Section 2.7.3.)

ACCEPTS IN AC1: Process handle

AC2: Length of the entry vector, or 0

AC3: Address at which the entry vector starts

TOPS-20 MONITOR CALLS
(XSVEC%)

RETURNS +1: Always

A zero in AC2 clears the process entry vector.

The XGVEC% monitor call can be used to obtain the entry vector of the process.

Generates an illegal instruction interrupt on error conditions below.

XSVEC% ERROR MNEMONICS:

FRKHX1: Invalid process handle
FRKHX2: Illegal to manipulate superior process
FRKHX3: Invalid use of multiple process handle
FRKHX8: Illegal to manipulate an execute-only process
SEVEX1: Entry vector length is not less than 1000

APPENDIX A

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

Table A-1 shows the ASCII and SIXBIT collating sequences and the conversions from ASCII to EBCDIC. If the ASCII character does not convert to the same character in EBCDIC, the EBCDIC character is shown in parentheses next to the EBCDIC code. Note that the first and last 32 characters do not exist in SIXBIT. Also, the characters in the first column of page A-1 (NUL, SOH, STX, and so forth) are control characters, which are nonprinting.

Table A-1: ASCII and SIXBIT Collating Sequence and Conversion to EBCDIC

| Character | ASCII 7-bit | EBCDIC 9-bit | Character | SIXBIT | ASCII 7-bit | EBCDIC 9-bit |
|-----------|----------------|-----------------|-----------|--------|----------------|-----------------|
| NUL | 000 | 000 | Space | 00 | 040 | 100 |
| SOH | 001 | 001* | ! | 01 | 041 | 132 |
| STX | 002 | 002* | " | 02 | 042 | 177 |
| ETX | 003 | 003* | # | 03 | 043 | 173 |
| EOT | 004 | 067 | \$ | 04 | 044 | 133 |
| ENQ | 005 | 055* | % | 05 | 045 | 154 |
| ACK | 006 | 056* | & | 06 | 046 | 120 |
| BEL | 007 | 057* | ' | 07 | 047 | 175 |
| BS | 010 | 026 | (| 10 | 050 | 115 |
| HT | 011 | 005 |) | 11 | 051 | 135 |
| LF | 012 | 045 | * | 12 | 052 | 134 |
| VT | 013 | 013* | + | 13 | 053 | 116 |
| FF | 014 | 014* | , | 14 | 054 | 153 |
| CR | 015 | 025*(NL) | - | 15 | 055 | 140 |
| SO | 016 | 006*(LC) | . | 16 | 056 | 113 |
| SI | 017 | 066*(UC) | / | 17 | 057 | 141 |
| DLE | 020 | 044*(BYP) | 0 | 20 | 060 | 360 |
| DC1 | 021 | 024*(RES) | 1 | 21 | 061 | 361 |
| DC2 | 022 | 064*(PN) | 2 | 22 | 062 | 362 |

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

| | | | | | | |
|-----|-----|-----------|---|----|-----|-----|
| DC3 | 023 | 065*(RS) | 3 | 23 | 063 | 363 |
| DC4 | 024 | 004*(PF) | 4 | 24 | 064 | 364 |
| NAK | 025 | 075* | 5 | 25 | 065 | 365 |
| SYN | 026 | 027*(IL) | 6 | 26 | 066 | 366 |
| ETB | 027 | 046*(EOB) | 7 | 27 | 067 | 367 |
| CAN | 030 | 052*(CM) | 8 | 30 | 070 | 370 |
| EM | 031 | 031* | 9 | 31 | 071 | 371 |
| SUB | 032 | 032*(CC) | : | 32 | 072 | 172 |
| ESC | 033 | 047*(PRE) | ; | 33 | 073 | 136 |
| FS | 034 | 023*(TM) | < | 34 | 074 | 114 |
| GS | 035 | 041*(SOS) | = | 35 | 075 | 176 |
| RS | 036 | 040*(DS) | > | 36 | 076 | 156 |
| US | 037 | 042*(FS) | ? | 37 | 077 | 157 |

| Character | SIXBIT | ASCII 7-bit | EBCDIC 9-bit | Character | ASCII 7-bit | EBCDIC 9-bit |
|-----------|--------|----------------|-----------------|-----------|----------------|-----------------|
| @ | 40 | 100 | 174 | a | 140 | 171 |
| A | 41 | 101 | 301 | b | 141 | 201 |
| B | 42 | 102 | 302 | c | 142 | 202 |
| C | 43 | 103 | 303 | d | 143 | 203 |
| D | 44 | 104 | 304 | e | 144 | 204 |
| E | 45 | 105 | 305 | f | 145 | 205 |
| F | 46 | 106 | 306 | g | 146 | 206 |
| G | 47 | 107 | 307 | h | 147 | 207 |
| H | 50 | 110 | 310 | i | 150 | 210 |
| I | 51 | 111 | 311 | j | 151 | 211 |
| J | 52 | 112 | 321 | k | 152 | 221 |
| K | 53 | 113 | 322 | l | 153 | 222 |
| L | 54 | 114 | 323 | m | 154 | 223 |
| M | 55 | 115 | 324 | n | 155 | 224 |
| N | 56 | 116 | 325 | o | 156 | 225 |
| O | 57 | 117 | 326 | p | 157 | 226 |
| P | 60 | 120 | 327 | q | 160 | 227 |
| Q | 61 | 121 | 330 | r | 161 | 230 |
| R | 62 | 122 | 331 | s | 162 | 231 |
| S | 63 | 123 | 342 | t | 163 | 242 |
| T | 64 | 124 | 343 | u | 164 | 243 |
| U | 65 | 125 | 344 | v | 165 | 244 |
| V | 66 | 126 | 345 | w | 166 | 245 |
| W | 67 | 127 | 346 | x | 167 | 246 |
| X | 70 | 130 | 347 | y | 170 | 247 |
| Y | 71 | 131 | 350 | z | 171 | 250 |
| Z | 72 | 132 | 351 | { | 172 | 251 |
| [| 73 | 133 | 255(1) | } | 173 | 300[1] |
| [1] | | | | | 174 | 117 |
| \ | 74 | 134 | 340 | } | 175 | 320 |
|] | 75 | 135 | 275 | | | |

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

| | | | | | | |
|---|----|-----|-----|--------|-----|-----|
| ^ | 76 | 136 | 137 | ~ | 176 | 241 |
| - | 77 | 137 | 155 | Delete | 177 | 007 |

Table A-2 shows the EBCDIC collating sequence and the conversion from EBCDIC to ASCII.

[1] [1] These EBCDIC codes either have no equivalent in the ASCII or SIXBIT character sets, or are referred to by different names. They are converted to the indicated ASCII characters to preserve their uniqueness if the ASCII character is converted back to EBCDIC.

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

Table A-2: EBCDIC Collating Sequence and Conversion to ASCII

| EBCDIC code | EBCDIC character | ASCII code | ASCII character | EBCDIC code | EBCDIC character | ASCII code | ASCII character |
|-------------|------------------|------------|-----------------|-------------|------------------|------------|-----------------|
| 000 | NUL | 000 | NUL | 060 | | 134 | \ |
| 001 | SOH | 001 | SOH | 061 | | 134 | \ |
| 002 | STX | 002 | STX | 062 | | 134 | \ |
| 003 | ETX | 003 | ETX | 063 | | 134 | \ |
| 004 | PF | 024 | DC4 | 064 | PN | 022 | DC2 |
| 005 | HT | 011 | HT | 065 | RS | 023 | DC3 |
| 006 | LC | 016 | SO | 066 | UC | 017 | SI |
| 007 | Delete | 177 | Delete | 067 | EOT | 004 | EOT |
| 010 | | 134 | \ | 070 | | 134 | \ |
| 011 | | 134 | \ | 071 | | 134 | \ |
| 012 | SMM | 134 | \ | 072 | | 134 | \ |
| 013 | VT | 013 | VT | 073 | | 134 | \ |
| 014 | FF | 014 | FF | 074 | CU3 | 134 | \ |
| 015 | CR | 134 | \ | 075 | DC4 | 025 | NAK |
| 016 | SO | 134 | \ | 076 | NAK | 134 | \ |
| 017 | SI | 134 | \ | 077 | SUB | 134 | \ |
| 020 | DLE | 134 | \ | 100 | Space | 040 | Space |
| 021 | DC1 | 134 | \ | 101 | | 134 | \ |
| 022 | DC2 | 134 | \ | 102 | | 134 | \ |
| 023 | TM | 034 | FS | 103 | | 134 | \ |
| 024 | RES | 021 | DC1 | 104 | | 134 | \ |
| 025 | NL | 015 | CR | 105 | | 134 | \ |
| 026 | BS | 010 | BS | 106 | | 134 | \ |
| 027 | IL | 026 | SYN | 107 | | 134 | \ |
| 030 | CAN | 134 | \ | 110 | | 134 | \ |
| 031 | EM | 031 | EM | 111 | | 134 | \ |
| 032 | CC | 032 | SUB | 112 | CENT | 134 | \ |
| 033 | CU1 | 134 | \ | 113 | . | 056 | . |
| 034 | IFS | 134 | \ | 114 | < | 074 | < |
| 035 | IGS | 134 | \ | 115 | (| 050 | (|
| 036 | IRS | 134 | \ | 116 | + | 053 | + |
| 037 | IUS | 134 | \ | 117 | | 174 | |

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

Table A-2: EBCDIC Collating Sequence and Conversion to ASCII (Cont.)

| EBCDIC code | EBCDIC character | ASCII code | ASCII character | EBCDIC code | EBCDIC character | ASCII code | ASCII character |
|-------------|------------------|------------|-----------------|-------------|------------------|------------|-----------------|
| 040 | DS | 036 | RS | 120 | & | 046 | & |
| 041 | SOS | 035 | GS | 121 | | 134 | \ |
| 042 | FS | 037 | US | 122 | | 134 | \ |
| 043 | | 134 | \ | 123 | | 134 | \ |
| 044 | BYP | 020 | DLE | 124 | | 134 | \ |
| 045 | LF | 012 | LF | 125 | | 134 | \ |
| 046 | ETB | 027 | ETB | 126 | | 134 | \ |
| 047 | ESC | 033 | ESC | 127 | | 134 | \ |
| 050 | | 134 | \ | 130 | | 134 | \ |
| 051 | | 134 | \ | 131 | | 134 | \ |
| 052 | SM | 030 | CAN | 132 | ! | 041 | ! |
| 053 | CUZ | 134 | \ | 133 | \$ | 044 | \$ |
| 054 | | 134 | \ | 134 | * | 052 | * |
| 055 | ENQ | 005 | ENQ | 135 |) | 051 |) |
| 056 | ACK | 006 | ACK | 136 | ^ | 073 | ^ |
| 057 | BEL | 007 | BEL | 137 | | 137 | \ |
| 140 | - | 055 | - | 220 | | 134 | \ |
| 141 | \ | 057 | / | 221 | j | 152 | j |
| 142 | | 134 | \ | 222 | k | 153 | k |
| 143 | | 134 | \ | 223 | l | 154 | l |
| 144 | | 134 | \ | 224 | m | 155 | m |
| 145 | | 134 | \ | 225 | n | 156 | n |
| 146 | | 134 | \ | 226 | o | 157 | o |
| 147 | | 134 | \ | 227 | p | 160 | p |
| 150 | | 134 | \ | 230 | q | 161 | q |
| 151 | | 134 | \ | 231 | r | 162 | r |
| 152 | | 134 | \ | 232 | | 134 | \ |
| 153 | , | 054 | , | 233 | | 134 | \ |
| 154 | % | 045 | % | 234 | | 134 | \ |
| 155 | | 137 | | 235 | | 134 | \ |
| 156 | > | 076 | > | 236 | | 134 | \ |
| 157 | ? | 077 | ? | 237 | | 134 | \ |

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

Table A-2: EBCDIC Collating Sequence and Conversion to ASCII (Cont.)

| EBCDIC code | EBCDIC character | ASCII code | ASCII character | EBCDIC code | EBCDIC character | ASCII code | ASCII character |
|-------------|------------------|------------|-----------------|-------------|------------------|------------|-----------------|
| 160 | | 134 | \ | 240 | | 134 | \ |
| 161 | | 134 | \ | 241 | | 176 | ~ |
| 162 | | 134 | \ | 242 | s | 163 | s |
| 163 | | 134 | \ | 243 | t | 164 | t |
| 164 | | 134 | \ | 244 | u | 165 | u |
| 165 | | 134 | \ | 245 | v | 166 | v |
| 166 | | 134 | \ | 246 | w | 167 | w |
| 167 | | 134 | \ | 247 | x | 170 | x |
| 170 | | 134 | \ | 250 | y | 171 | y |
| 171 | | 140 | | 251 | z | 172 | z |
| 172 | : | 072 | : | 252 | | 134 | \ |
| 173 | # | 043 | # | 253 | | 134 | \ |
| 174 | @ | 100 | @ | 254 | | 134 | \ |
| 175 | ' | 47 | ' | 255 | [| 133 | [|
| 176 | = | 075 | = | 256 | | 134 | \ |
| 177 | " | 042 | " | 257 | | 134 | \ |
| 200 | | 134 | \ | 260 | | 175 | |
| 201 | a | 141 | a | 261 | | 134 | \ |
| 202 | b | 142 | b | 262 | | 134 | \ |
| 203 | c | 143 | c | 263 | | 134 | \ |
| 204 | d | 144 | d | 264 | | 134 | \ |
| 205 | e | 145 | e | 265 | | 134 | \ |
| 206 | f | 146 | f | 266 | | 134 | \ |
| 207 | g | 147 | g | 267 | | 134 | \ |
| 210 | h | 150 | h | 270 | | 134 | \ |
| 211 | i | 151 | i | 271 | | 134 | \ |
| 212 | | 134 | \ | 272 | | 134 | \ |
| 213 | | 134 | \ | 273 | | 134 | \ |
| 214 | | 134 | \ | 274 | | 134 | \ |
| 215 | | 134 | \ | 275 |] | 135 |] |
| 216 | | 134 | \ | 276 | | 134 | \ |
| 217 | | 134 | \ | 277 | | 134 | \ |

ASCII, SIXBIT, AND EBCDIC COLLATING SEQUENCES AND CONVERSIONS

Table A-2: EBCDIC Collating Sequence and Conversion to ASCII (Cont.)

| EBCDIC code | EBCDIC character | ASCII code | ASCII character | EBCDIC code | EBCDIC character | ASCII code | ASCII character |
|-------------|------------------|------------|-----------------|-------------|------------------|------------|-----------------|
| 300 | | 173 | { | 340 | | 134 | \ |
| 301 | A | 101 | A | 341 | | 134 | \ |
| 302 | B | 102 | B | 342 | S | 123 | S |
| 303 | C | 103 | C | 343 | T | 124 | T |
| 304 | D | 104 | D | 344 | U | 125 | U |
| 305 | E | 105 | E | 345 | V | 126 | V |
| 306 | F | 106 | F | 346 | W | 127 | W |
| 307 | G | 107 | G | 347 | X | 130 | X |
| 310 | H | 110 | H | 350 | Y | 131 | Y |
| 311 | I | 110 | I | 351 | Z | 132 | Z |
| 312 | | 134 | \ | 352 | | 134 | \ |
| 313 | | 134 | \ | 353 | | 134 | \ |
| 314 | | 134 | \ | 354 | | 134 | \ |
| 315 | | 134 | \ | 355 | | 134 | \ |
| 316 | | 134 | \ | 356 | | 134 | \ |
| 317 | | 134 | \ | 357 | | 134 | \ |
| 320 | | 175 | } | 360 | 0 | 060 | 1 |
| 321 | J | 112 | J | 361 | 1 | 061 | 1 |
| 322 | K | 113 | K | 362 | 2 | 062 | 2 |
| 323 | L | 114 | L | 363 | 3 | 063 | 3 |
| 324 | M | 115 | M | 364 | 4 | 064 | 4 |
| 325 | N | 116 | N | 365 | 5 | 065 | 5 |
| 326 | O | 117 | O | 366 | 6 | 066 | 6 |
| 327 | P | 120 | P | 367 | 7 | 067 | 7 |
| 330 | Q | 121 | Q | 370 | 8 | 070 | 8 |
| 331 | R | 122 | R | 371 | 9 | 071 | 9 |
| 332 | | 134 | \ | 372 | | 134 | \ |
| 333 | | 134 | \ | 373 | | 134 | \ |
| 334 | | 134 | \ | 374 | | 134 | \ |
| 335 | | 134 | \ | 375 | | 134 | \ |
| 336 | | 134 | \ | 376 | | 134 | \ |
| 337 | | 134 | \ | 377 | | 134 | \ |

APPENDIX B

TOPS-20 ERROR CODES AND MNEMONICS

| Code | Mnemonic | Code | Mnemonic | Code | Mnemonic |
|--------|----------|--------|----------|--------|----------|
| 600010 | LGINX1 | 600011 | LGINX2 | 600012 | LGINX3 |
| 600013 | LGINX4 | 600014 | LGINX5 | 600020 | CRJBX1 |
| 600021 | CRJBX2 | 600022 | CRJBX3 | 600023 | CRJBX4 |
| 600024 | CRJBX5 | 600025 | CRJBX6 | 600026 | CRJBX7 |
| 600035 | LOUTX1 | 600036 | LOUTX2 | 600045 | CACTX1 |
| 600046 | CACTX2 | 600050 | EFCTX1 | 600051 | EFCTX2 |
| 600052 | EFCTX3 | 600055 | GJFX1 | 600056 | GJFX2 |
| 600057 | GJFX3 | 600060 | GJFX4 | 600061 | GJFX5 |
| 600062 | GJFX6 | 600063 | GJFX7 | 600064 | GJFX8 |
| 600065 | GJFX9 | 600066 | GJFX10 | 600067 | GJFX11 |
| 600070 | GJFX12 | 600071 | GJFX13 | 600072 | GJFX14 |
| 600073 | GJFX15 | 600074 | GJFX16 | 600075 | GJFX17 |
| 600076 | GJFX18 | 600077 | GJFX19 | 600100 | GJFX20 |
| 600101 | GJFX21 | 600102 | GJFX22 | 600103 | GJFX23 |
| 600104 | GJFX24 | 600107 | GJFX27 | 600110 | GJFX28 |
| 600111 | GJFX29 | 600112 | GJFX30 | 600113 | GJFX31 |
| 600114 | GJFX32 | 600115 | GJFX33 | 600116 | GJFX34 |
| 600117 | GJFX35 | 600120 | OPNX1 | 600121 | OPNX2 |
| 600122 | OPNX3 | 600123 | OPNX4 | 600124 | OPNX5 |
| 600125 | OPNX6 | 600126 | OPNX7 | 600127 | OPNX8 |
| 600130 | OPNX9 | 600131 | OPNX10 | 600133 | OPNX12 |
| 600134 | OPNX13 | 600135 | OPNX14 | 600136 | OPNX15 |
| 600137 | OPNX16 | 600140 | OPNX17 | 600141 | OPNX18 |
| 600142 | OPNX19 | 600143 | OPNX20 | 600144 | OPNX21 |
| 600145 | OPNX22 | 600150 | DESX1 | 600151 | DESX2 |
| 600152 | DESX3 | 600153 | DESX4 | 600154 | DESX5 |
| 600155 | DESX6 | 600156 | DESX7 | 600157 | DESX8 |
| 600160 | CLSX1 | 600161 | CLSX2 | 600165 | RJFNX1 |
| 600166 | RJFNX2 | 600167 | RJFNX3 | 600170 | DELFX1 |
| 600175 | SFPTX1 | 600176 | SFPTX2 | 600177 | SFPTX3 |
| 600200 | CNDIX1 | 600202 | CNDIX3 | 600204 | CNDIX5 |
| 600210 | SFBSX1 | 600211 | SFBSX2 | 600215 | IOX1 |
| 600216 | IOX2 | 600217 | IOX3 | 600220 | IOX4 |
| 600221 | IOX5 | 600222 | IOX6 | 600240 | PMAPX1 |
| 600241 | PMAPX2 | 600245 | SPACX1 | 600250 | FRKHX1 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 600251 | FRKHX2 | 600252 | FRKHX3 | 600253 | FRKHX4 |
| 600254 | FRKHX5 | 600255 | FRKHX6 | 600260 | SPLFX1 |
| 600261 | SPLFX2 | 600262 | SPLFX3 | 600263 | SPLBTS |
| 600264 | SPLBFC | 600267 | GTABX1 | 600270 | GTABX2 |
| 600271 | GTABX3 | 600273 | RUNTX1 | 600275 | STADX1 |
| 600276 | STADX2 | 600300 | ASNDX1 | 600301 | ASNDX2 |
| 600302 | ASNDX3 | 600320 | ATACX1 | 600321 | ATACX2 |
| 600322 | ATACX3 | 600323 | ATACX4 | 600324 | ATACX5 |
| 600332 | STDVX1 | 600335 | DEVX1 | 600336 | DEVX2 |
| 600337 | DEVX3 | 600345 | MNTX1 | 600346 | MNTX2 |
| 600347 | MNTX3 | 600350 | TERMX1 | 600351 | TLNKX1 |
| 600352 | ATIX1 | 600353 | ATIX2 | 600356 | TLNKX2 |
| 600357 | TLNKX3 | 600360 | TTYX1 | 600361 | RSCNX1 |
| 600362 | RSCNX2 | 600363 | CFRKX3 | 600365 | KFRKX1 |
| 600366 | KFRKX2 | 600367 | RFRKX1 | 600370 | HFRKX1 |
| 600371 | GFRKX1 | 600373 | GETX1 | 600374 | GETX2 |
| 600375 | TFRKX1 | 600376 | TFRKX2 | 600377 | SFRVX1 |
| 600407 | NOUTX1 | 600410 | NOUTX2 | 600411 | TFRKX3 |
| 600414 | IFIXX1 | 600415 | IFIXX2 | 600416 | IFIXX3 |
| 600424 | GFDBX1 | 600425 | GFDBX2 | 600426 | GFDBX3 |
| 600430 | CFDBX1 | 600431 | CFDBX2 | 600432 | CFDBX3 |
| 600433 | CFDBX4 | 600434 | CFDBX5 | 600440 | DUMPX1 |
| 600441 | DUMPX2 | 600442 | DUMPX3 | 600443 | DUMPX4 |
| 600450 | RNAMX1 | 600451 | RNAMX2 | 600452 | RNAMX3 |
| 600453 | RNAMX4 | 600454 | BKJFX1 | 600460 | TIMEX1 |
| 600461 | ZONEX1 | 600462 | ODTNX1 | 600464 | DILFX1 |
| 600465 | TILFX1 | 600466 | DATEX1 | 600467 | DATEX2 |
| 600470 | DATEX3 | 600471 | DATEX4 | 600472 | DATEX5 |
| 600473 | DATEX6 | 600516 | SMONX1 | 600517 | MSCPX1 |
| 600520 | MSCPX2 | 600521 | MSCPX3 | 600530 | SACTX1 |
| 600531 | SACTX2 | 600532 | SACTX3 | 600533 | SACTX4 |
| 600540 | GACTX1 | 600541 | GACTX2 | 600544 | FFUFX1 |
| 600545 | FFUFX2 | 600546 | FFUFX3 | 600555 | DSMX1 |
| 600560 | RDDIX1 | 600570 | SIRX1 | 600600 | SSAVX1 |
| 600601 | SSAVX2 | 600610 | SEVEX1 | 600614 | WHELX1 |
| 600615 | CAPX1 | 600617 | PEEKX2 | 600620 | CRDIX1 |
| 600621 | CRDIX2 | 600622 | CRDIX3 | 600623 | CRDIX4 |
| 600624 | CRDIX5 | 600626 | CRDIX7 | 600640 | CRDIX1 |
| 600641 | GTDIX2 | 600650 | FLINX1 | 600651 | FLINX2 |
| 600652 | FLINX3 | 600653 | FLINX4 | 600660 | FLOTX1 |
| 600661 | FLOTX2 | 600662 | FLOTX3 | 600670 | HPTX1 |
| 600700 | FDFRX1 | 600701 | FDFRX2 | 600703 | GTHSX1 |
| 600704 | GTHSX2 | 600705 | GTHSX3 | 600706 | GTHSX4 |
| 600707 | GTHSX5 | 600710 | ATNX1 | 600711 | ATNX2 |
| 600712 | ATNX3 | 600713 | ATNX4 | 600714 | ATNX5 |
| 600715 | ATNX6 | 600716 | ATNX7 | 600717 | ATNX8 |
| 600720 | ATNX9 | 600721 | ATNX10 | 600722 | ATNX11 |
| 600723 | ATNX12 | 600724 | ATNX13 | 600727 | CVHST1 |
| 600730 | CVSKX1 | 600731 | CVSKX2 | 600732 | SNDIX1 |
| 600733 | SNDIX2 | 600734 | SNDIX3 | 600735 | SNDIX4 |
| 600736 | SNDIX5 | 600737 | NTWZX1 | 600740 | ASNSX1 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 600741 | ASNSX2 | 600742 | SQX1 | 600743 | SQX2 |
| 600746 | GTNCX1 | 600747 | GTNCX2 | 600750 | RNAMX5 |
| 600751 | RNAMX6 | 600752 | RNAMX7 | 600753 | RNAMX8 |
| 600754 | RNAMX9 | 600755 | RNMX10 | 600756 | RNMX11 |
| 600757 | RNMX12 | 600760 | GJFX36 | 600770 | ILINS1 |
| 600771 | ILINS2 | 600772 | ILINS3 | 601000 | CRLNX1 |
| 601001 | INLNX1 | 601002 | LNSTX1 | 601003 | MLKBX1 |
| 601004 | MLKBX2 | 601005 | MLKBX3 | 601006 | MLKBX4 |
| 601007 | VBCX1 | 601010 | RDTX1 | 601011 | GFKSX1 |
| 601013 | GTJIX1 | 601014 | GTJIX2 | 601015 | GTJIX3 |
| 601016 | IPCFX1 | 601017 | IPCFX2 | 601020 | IPCFX3 |
| 601021 | IPCFX4 | 601022 | IPCFX5 | 601023 | IPCFX6 |
| 601024 | IPCFX7 | 601025 | IPCFX8 | 601026 | IPCFX9 |
| 601027 | IPCF10 | 601030 | IPCF11 | 601031 | IPCF12 |
| 601032 | IPCF13 | 601033 | IPCF14 | 601034 | IPCF15 |
| 601035 | IPCF16 | 601036 | IPCF17 | 601037 | IPCF18 |
| 601040 | IPCF19 | 601041 | IPCF20 | 601042 | IPCF21 |
| 601043 | IPCF22 | 601044 | IPCF23 | 601045 | IPCF24 |
| 601046 | IPCF25 | 601047 | IPCF26 | 601050 | IPCF27 |
| 601051 | IPCF28 | 601052 | IPCF29 | 601053 | IPCF30 |
| 601054 | GNJFX1 | 601055 | ENQX1 | 601056 | ENQX2 |
| 601057 | ENQX3 | 601060 | ENQX4 | 601061 | ENQX5 |
| 601062 | ENQX6 | 601063 | ENQX7 | 601064 | ENQX8 |
| 601065 | ENQX9 | 601066 | ENQX10 | 601067 | ENQX11 |
| 601070 | ENQX12 | 601071 | ENQX13 | 601072 | ENQX14 |
| 601073 | ENQX15 | 601074 | ENQX16 | 601075 | ENQX17 |
| 601076 | ENQX18 | 601077 | ENQX19 | 601100 | ENQX20 |
| 601101 | ENQX21 | 601102 | IPCF31 | 601103 | IPCF32 |
| 601104 | PMAPX3 | 601105 | PMAPX4 | 601106 | PMAPX5 |
| 601107 | PMAPX6 | 601110 | SNOPX1 | 601111 | SNOPX2 |
| 601112 | SNOPX3 | 601113 | SNOPX4 | 601114 | SNOPX5 |
| 601115 | SNOPX6 | 601116 | SNOPX7 | 601117 | SNOPX8 |
| 601120 | SNOPX9 | 601121 | SNOP10 | 601122 | SNOP11 |
| 601123 | SNOP12 | 601124 | SNOP13 | 601125 | SNOP14 |
| 601126 | SNOP15 | 601127 | SNOP16 | 601130 | IPCF33 |
| 601131 | SNOP17 | 601132 | OPNX23 | 601133 | GJFX37 |
| 601134 | CRLNX2 | 601135 | INLNX2 | 601136 | LNSTX2 |
| 601137 | ALCX1 | 601140 | ALCX2 | 601141 | ALCX3 |
| 601142 | ALCX4 | 601143 | ALCX5 | 601144 | SPLX1 |
| 601145 | SPLX2 | 601146 | SPLX3 | 601147 | SPLX4 |
| 601150 | SPLX5 | 601151 | CLSX3 | 601152 | CRLNX3 |
| 601153 | ALCX6 | 601154 | CKAX1 | 601155 | CKAX2 |
| 601156 | CKAX3 | 601157 | TIMX1 | 601160 | TIMX2 |
| 601161 | TIMX3 | 601162 | TIMX4 | 601163 | SNOP18 |
| 601164 | GJFX38 | 601165 | GJFX39 | 601166 | CRDIX8 |
| 601167 | CRDIX9 | 601170 | CRDI10 | 601171 | DELDX1 |
| 601172 | DELDX2 | 601173 | GACTX3 | 601174 | DIAGX1 |
| 601175 | DIAGX2 | 601176 | DIAGX3 | 601177 | DIAGX4 |
| 601200 | DIAGX5 | 601201 | DIAGX6 | 601202 | DIAGX7 |
| 601203 | DIAGX8 | 601204 | DIAGX9 | 601205 | DIAG10 |
| 601206 | SYEX1 | 601207 | SYEX2 | 601210 | MTOX1 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 601211 | IOX7 | 601212 | IOX8 | 601213 | MTOX5 |
| 601214 | DUMPX5 | 601215 | DUMPX6 | 601216 | IOX9 |
| 601217 | CLSX4 | 601220 | MTOX2 | 601221 | MTOX3 |
| 601222 | MTOX4 | 601223 | MTOX6 | 601224 | OPNX25 |
| 601225 | GJFX40 | 601226 | MTOX7 | 601227 | LOUTX3 |
| 601230 | LOUTX4 | 601231 | CAPX2 | 601232 | SSAVX3 |
| 601233 | SSAVX4 | 601234 | TDELX1 | 601235 | TADDX1 |
| 601236 | TADDX2 | 601237 | TLUKX1 | 601240 | IOX10 |
| 601241 | CNDIX2 | 601242 | CNDIX4 | 601243 | CNDIX6 |
| 601244 | SJBX1 | 601245 | SJBX2 | 601246 | SJBX3 |
| 601247 | TMONX1 | 601250 | SMONX2 | 601251 | SJBX4 |
| 601252 | SJBX5 | 601253 | SJBX6 | 601254 | GTJIX4 |
| 601255 | ILINS4 | 601256 | ILINS5 | 601257 | COMNX1 |
| 601260 | COMNX2 | 601261 | COMNX3 | 601262 | COMNX4 |
| 601263 | PRAX1 | 601264 | PRAX2 | 601265 | COMNX5 |
| 601266 | COMNX6 | 601267 | COMNX7 | 601270 | PRAX3 |
| 601271 | CKAX4 | 601272 | GACCX1 | 601273 | GACCX2 |
| 601274 | MTOX8 | 601275 | DBRXX1 | 601276 | SJPRX1 |
| 601277 | GJFX41 | 601300 | GJFX42 | 601301 | GACCX3 |
| 601302 | TIMEX2 | 601303 | DELFX2 | 601304 | DELFX3 |
| 601305 | DELFX4 | 601306 | DELFX5 | 601307 | DELFX6 |
| 601310 | DELFX7 | 601311 | DELFX8 | 601312 | FRKHX7 |
| 601313 | DIRX1 | 601314 | DIRX2 | 601315 | DIRX3 |
| 601316 | UFPGX1 | 601317 | LNGFX1 | 601320 | IPCF34 |
| 601321 | COMNX8 | 601322 | MTOX9 | 601323 | MTOX10 |
| 601324 | MTOX11 | 601325 | MTOX12 | 601326 | MTOX13 |
| 601327 | MTOX14 | 601330 | SAVX1 | 601331 | MTOX15 |
| 601332 | MTOX16 | 601333 | LPINX1 | 601334 | LPINX2 |
| 601335 | LPINX3 | 601336 | MTOX17 | 601337 | LGINX6 |
| 601340 | DESX9 | 601341 | ACESX1 | 601342 | ACESX2 |
| 601343 | DSKX1 | 601344 | DSKX2 | 601345 | MSTRX1 |
| 601346 | MSTRX2 | 601347 | MSTRX3 | 601350 | MSTRX4 |
| 601351 | MSTRX5 | 601352 | MSTRX6 | 601353 | MSTRX7 |
| 601354 | MSTRX8 | 601355 | MSTRX9 | 601356 | MSTX10 |
| 601357 | MSTX11 | 601360 | MSTX12 | 601361 | MSTX13 |
| 601362 | MSTX14 | 601363 | MSTX15 | 601364 | MSTX16 |
| 601365 | DSKX01 | 601366 | DSKX02 | 601367 | DSKX03 |
| 601370 | DSKX04 | 601371 | GFUSX1 | 601372 | GFUSX2 |
| 601373 | SFUSX1 | 601374 | SFUSX2 | 601375 | SFUSX3 |
| 601376 | RCDIX1 | 601377 | RCDIX2 | 601400 | RCDIX3 |
| 601401 | RCDIX4 | 601402 | RCUSX1 | 601403 | TDELX2 |
| 601404 | TIMX5 | 601405 | LSTRX1 | 601406 | SWJFX1 |
| 601407 | MTOX18 | 601410 | OPNX26 | 601411 | DELFX9 |
| 601412 | CRDIX6 | 601413 | COMNX9 | 601414 | STYPX1 |
| 601415 | PMAPX7 | 601416 | DSKX3 | 601417 | DESX10 |
| 601420 | DSKX4 | 601421 | MSTX17 | 601422 | MSTX18 |
| 601423 | MSTX19 | 601424 | MSTX20 | 601425 | MSTX21 |
| 601426 | MSTX22 | 601427 | CRDI11 | 601430 | MSTX23 |
| 601431 | ACESX3 | 601432 | ACESX4 | 601433 | ACESX5 |
| 601434 | STRX05 | 601435 | ACESX6 | 601436 | STRX01 |
| 601437 | STRX02 | 601440 | IOX11 | 601441 | IOX12 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 601442 | STRX03 | 601443 | STRX04 | 601444 | PPNX1 |
| 601445 | PPNX2 | 601446 | PPNX3 | 601447 | PPNX4 |
| 601450 | SPLX6 | 601451 | CRDI12 | 601452 | GFUSX3 |
| 601453 | GFUSX4 | 601454 | RNMX13 | 601455 | SJBX8 |
| 601456 | DECRSV | 601457 | FFFFX1 | 601460 | WILDY1 |
| 601461 | MSTX41 | 601462 | MSTX42 | 601463 | CIMXND |
| 601464 | CINOND | 601465 | CIBDOF | 601466 | CINOFQ |
| 601467 | CINOPG | 601470 | CINPTH | 601471 | CIBDCD |
| 601472 | CIUNOP | 601473 | CINOND | 601474 | CILNER |
| 601475 | LCBDBP | 601476 | LCLNER | 601477 | LCNOND |
| 601500 | SSAVX5 | 601501 | CIBDFQ | 601502 | ATACX6 |
| 601503 | ATACX7 | 601504 | QUEUX1 | 601505 | QUEUX2 |
| 601506 | QUEUX3 | 601507 | QUEUX4 | 601510 | QUEUX5 |
| 601511 | QUEUX6 | 601512 | QUEUX7 | 601513 | DIAG21 |
| 601514 | MTNX01 | 601515 | DIAG22 | 601516 | DIAG23 |
| 601517 | DIAG24 | 601520 | DIAG25 | 601521 | DIAG26 |
| 601522 | DIAG27 | 601523 | DIAG30 | 601524 | SCSTBF |
| 601525 | MSTX47 | 601526 | MSTX48 | 601527 | MSTX49 |
| 601530 | PAGPTN | 601531 | MSTX50 | 601532 | MSTX51 |
| 601533 | DSKOX5 | 601534 | DSKOX6 | 601535 | TIMX6 |
| 601536 | TIMX7 | 601537 | TIMX8 | 601540 | TIMX9 |
| 601541 | TIMX10 | 601550 | SCTX1 | 601551 | SCTX2 |
| 601552 | SCTX3 | 601553 | SCTX4 | 601554 | PDVX01 |
| 601555 | PDVX02 | 601556 | PDVX03 | 601557 | GETX4 |
| 601560 | GETX5 | 601561 | DYNX01 | 601562 | DYNX02 |
| 601563 | DYNX03 | 601564 | DYNX04 | 601565 | DYNX05 |
| 601566 | DYNX06 | 601567 | DYNX07 | 601570 | DYNX08 |
| 601571 | DYNX09 | 601572 | DYNX10 | 601573 | DYNX11 |
| 601600 | CTSX01 | 601601 | CTSX02 | 601602 | CTSX03 |
| 601603 | CTSX04 | 601604 | CTSX05 | 601605 | CTSX06 |
| 601606 | CTSX07 | 601607 | CTSX08 | 601610 | CTSX09 |
| 601611 | CTSX10 | 601612 | CTSX11 | 601613 | CTSX12 |
| 601614 | CTSX13 | 601615 | DOBX01 | 601616 | DOBX02 |
| 601617 | DOBX03 | 601620 | DOBX04 | 601621 | DOBX05 |
| 601622 | DOBX06 | 601623 | DOBX07 | 601624 | DOBX08 |
| 601674 | STRX11 | 601675 | USGX04 | 601676 | STRX10 |
| 601677 | SMONX3 | 601700 | SFUSX4 | 601701 | SFUSX5 |
| 601702 | SFUSX6 | 601703 | GETX3 | 601704 | FILX01 |
| 601705 | ARGX01 | 601706 | CAPX3 | 601707 | CAPX4 |
| 601711 | CAPX6 | 601712 | CAPX7 | 601713 | ARGX02 |
| 601714 | ARGX03 | 601715 | ARGX04 | 601716 | ARGX05 |
| 601717 | ARGX06 | 601720 | ARGX07 | 601721 | ARGX08 |
| 601722 | ARGX09 | 601723 | ARGX10 | 601724 | ARGX11 |
| 601725 | ARGX12 | 601726 | ARGX13 | 601727 | MONX01 |
| 601730 | MONX02 | 601731 | MONX03 | 601732 | MONX04 |
| 601733 | ARGX14 | 601734 | ARGX15 | 601735 | FILX02 |
| 601736 | FILX03 | 601737 | DEVX4 | 601740 | FILX04 |
| 601741 | ARGX16 | 601742 | ARGX17 | 601743 | ARGX18 |
| 601744 | DEVX5 | 601745 | DIRX4 | 601746 | FILX05 |
| 601747 | STRX06 | 601750 | MSTX24 | 601751 | MSTX25 |
| 601752 | MSTX26 | 601753 | LOUTX5 | 601754 | GJFX43 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 601755 | MTOX19 | 601756 | MTOX20 | 601757 | MSTX27 |
| 601760 | MSTX28 | 601761 | MSTX29 | 601763 | DSKX05 |
| 601764 | DSKX06 | 601765 | DSKX07 | 601766 | DSKX08 |
| 601767 | COMX10 | 601770 | MSTX30 | 601771 | LOCKX1 |
| 601772 | LOCKX2 | 601773 | LOCKX3 | 601774 | ILLX01 |
| 601775 | ILLX02 | 601776 | ILLX03 | 601777 | ILLX04 |
| 602000 | MSTX31 | 602001 | MSTX32 | 602002 | MSTX33 |
| 602003 | STDIX1 | 602004 | CNDIX7 | 602005 | PMCLX1 |
| 602006 | PMCLX2 | 602007 | PMCLX3 | 602010 | DLFX10 |
| 602011 | DLFX11 | 602012 | GJFX44 | 602013 | UTSTX1 |
| 602014 | UTSTX2 | 602015 | UTSTX3 | 602016 | BOTX01 |
| 602017 | BOTX02 | 602020 | DCNX1 | 602021 | DCNX5 |
| 602022 | DCNX3 | 602023 | DCNX4 | 602024 | DCNX9 |
| 602025 | DCNX8 | 602026 | DCNX11 | 602027 | DCNX12 |
| 602030 | TTYX01 | 602031 | BOTX03 | 602032 | MONX05 |
| 602033 | ARGX19 | 602035 | COMX11 | 602036 | COMX12 |
| 602037 | COMX13 | 602040 | COMX14 | 602041 | COMX15 |
| 602042 | COMX16 | 602043 | COMX17 | 602044 | NPXAMB |
| 602045 | NPXNSW | 602046 | NPXNOM | 602047 | NPXNUL |
| 602050 | NPXINW | 602051 | NPXNC | 602052 | NPXICN |
| 602053 | NPXIDT | 602054 | NPXNQS | 602055 | NPXNMT |
| 602056 | NPXNMD | 602057 | NPXCMA | 602060 | GJFX45 |
| 602061 | GJFX46 | 602062 | GJFX47 | 602063 | MSTX34 |
| 602064 | GJFX48 | 602065 | GJFX49 | 602077 | SJBX7 |
| 602100 | DELF10 | 602101 | CRDI13 | 602102 | CRDI14 |
| 602103 | CRDI15 | 602104 | CRDI16 | 602105 | ENACX1 |
| 602106 | ENACX2 | 602107 | ENACX3 | 602110 | ENACX4 |
| 602111 | VACCX0 | 602112 | VACCX1 | 602113 | USGX01 |
| 602114 | BOTX04 | 602115 | NODX01 | 602116 | USGX02 |
| 602117 | CRDI17 | 602120 | ENQX23 | 602121 | ENQX22 |
| 602122 | DCNX2 | 602123 | ABRXX1 | 602124 | USGX03 |
| 602125 | IPCF35 | 602126 | VACCX2 | 602127 | CRDI18 |
| 602130 | CRDI19 | 602131 | ENACX5 | 602132 | BOTX05 |
| 602133 | CRDI20 | 602134 | COMX18 | 602135 | COMX19 |
| 602136 | CRDI21 | 602137 | ACESX7 | 602140 | CRDI22 |
| 602141 | CRDI23 | 602142 | STRX07 | 602143 | STRX08 |
| 602144 | CRDI24 | 602146 | ATSX01 | 602147 | ATSX02 |
| 602150 | ATSX03 | 602151 | ATSX04 | 602152 | ATSX05 |
| 602153 | ATSX06 | 602154 | ATSX07 | 602155 | ATSX08 |
| 602156 | ATSX09 | 602157 | ATSX10 | 602160 | ATSX11 |
| 602161 | ATSX12 | 602162 | ATSX13 | 602163 | ATSX14 |
| 602164 | ATSX15 | 602165 | PMCLX4 | 602166 | ATSX16 |
| 602167 | ATSX17 | 602170 | FRKHX8 | 602171 | ARGX20 |
| 602172 | ARGX21 | 602173 | ARGX22 | 602174 | ATSX18 |
| 602175 | ATSX19 | 602176 | ATSX20 | 602177 | ARGX23 |
| 602200 | ARGX24 | 602201 | MSTX35 | 602202 | DCNX13 |
| 602203 | DCNX14 | 602204 | DCNX15 | 602205 | GJFX50 |
| 602206 | KDPX01 | 602207 | NODX02 | 602210 | NODX03 |
| 602211 | GJFX51 | 602212 | COMX20 | 602213 | ATSX21 |
| 602214 | ATSX22 | 602215 | ATSX23 | 602216 | ATSX24 |
| 602217 | ATSX25 | 602220 | GOKER1 | 602221 | GOKER2 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|---------|--------|---------|--------|---------|
| 602222 | STRX09 | 602223 | MSTX36 | 602224 | MSTX37 |
| 602225 | MSTX40 | 602226 | AT SX26 | 602227 | IOX13 |
| 602230 | IOX14 | 602231 | IOX15 | 602232 | IOX16 |
| 602233 | IOX17 | 602234 | IOX20 | 602235 | IOX21 |
| 602236 | IOX22 | 602237 | IOX23 | 602240 | IOX24 |
| 602241 | IOX25 | 602242 | SWJFX2 | 602243 | IOX26 |
| 602244 | IOX27 | 602245 | IOX30 | 602246 | ARGX25 |
| 602247 | SKDX1 | 602250 | MREQX1 | 602251 | MREQX2 |
| 602252 | MREQX3 | 602253 | MREQX4 | 602254 | MREQX5 |
| 602255 | MREQX6 | 602256 | MREQX7 | 602257 | MREQX8 |
| 602260 | MREQX9 | 602261 | MREQ10 | 602262 | MREQ11 |
| 602263 | MREQ12 | 602264 | MREQ13 | 602265 | MREQ14 |
| 602266 | MREQ15 | 602267 | MREQ16 | 602270 | MREQ17 |
| 602271 | MREQ18 | 602272 | MREQ19 | 602273 | MREQ20 |
| 602274 | MREQ21 | 602275 | DEVX6 | 602276 | AT SX27 |
| 602277 | AT SX28 | 602300 | AT SX29 | 602301 | AT SX30 |
| 602302 | AT SX31 | 602303 | AT SX32 | 602304 | AT SX33 |
| 602305 | AT SX34 | 602306 | AT SX35 | 602307 | AT SX36 |
| 602310 | DATEX7 | 602311 | MREQ22 | 602312 | ARCFX2 |
| 602313 | ARCFX3 | 602314 | ARCFX4 | 602315 | ARCFX5 |
| 602316 | ARCFX6 | 602317 | ARCFX7 | 602320 | ARCFX8 |
| 602321 | ARCFX9 | 602322 | ARCX10 | 602323 | ARCX11 |
| 602324 | ARCX12 | 602325 | ARCX13 | 602326 | OPNX30 |
| 602327 | OPNX31 | 602330 | DELX11 | 602331 | DELX12 |
| 602332 | ARCX14 | 602333 | ARCX15 | 602334 | ARCX16 |
| 602335 | ARCX17 | 602336 | ARCX18 | 602337 | ARCX19 |
| 602340 | ARGX26 | 602341 | ARGX27 | 602342 | DIRX5 |
| 602343 | IOX31 | 602344 | MREQ23 | 602345 | MREQ24 |
| 602346 | MREQ25 | 602347 | LTLBLX | 602350 | LTLBX1 |
| 602351 | MREQ26 | 602352 | METRX1 | 602353 | NSPX00 |
| 602354 | NSPX01 | 602355 | NSPX02 | 602356 | NSPX03 |
| 602357 | NSPX04 | 602360 | NSPX05 | 602361 | NSPX06 |
| 602362 | NSPX07 | 602363 | NSPX08 | 602364 | NSPX09 |
| 602365 | NSPX10 | 602366 | NSPX11 | 602367 | NSPX12 |
| 602370 | NSPX13 | 602371 | NSPX14 | 602372 | NSPX15 |
| 602373 | NSPX16 | 602374 | NSPX17 | 602375 | NSPX18 |
| 602376 | NSPX19 | 602377 | NSPX20 | 602400 | NSPX21 |
| 602401 | NSPX22 | 602402 | MREQ27 | 602403 | MREQ28 |
| 602404 | MREQ29 | 602405 | MREQ30 | 602406 | DIAG11 |
| 602407 | DIAG12 | 602410 | DESX11 | 602411 | NSPX23 |
| 602412 | ARGX28 | 602413 | NPX2CL | 602414 | ARGX29 |
| 602415 | ARGX30 | 602416 | ARGX31 | 602417 | DEVX7 |
| 602420 | GJFX52 | 602421 | GOKER3 | 602422 | IOX32 |
| 602423 | IOX33 | 602424 | XSIRX1 | 602425 | SIRX2 |
| 602426 | RIRX1 | 602427 | XSIRX2 | 602430 | MREQ31 |
| 602431 | SMAPX1 | 602432 | TTMSX1 | 602433 | MONX06 |
| 602434 | BOTX06 | 602435 | BOTX07 | 602436 | BOTX08 |
| 602437 | BOTX09 | 602440 | BOTX10 | 602441 | BOTX11 |
| 602442 | BOTX12 | 602443 | BOTX13 | 602444 | BOTX14 |
| 602445 | BOTX15 | 602446 | BOTX16 | 602447 | BOTX17 |
| 602450 | BOTX18 | 602451 | NTMX1 | 602452 | COMX21 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 602453 | DELX13 | 602454 | ANTX01 | 602455 | TTYX02 |
| 602456 | NSPX24 | 602457 | NSPX25 | 602460 | NSPX26 |
| 602461 | GJFX53 | 602462 | IOX34 | 602463 | IOX35 |
| 602464 | PMAPX8 | 602465 | SMAPX2 | 602466 | GJFX54 |
| 602467 | BOTX19 | 602470 | BOTX20 | 602471 | ILLX05 |
| 602472 | XSEVX1 | 602473 | XSEVX2 | 602474 | XSEVX3 |
| 602475 | ABRKX2 | 602476 | ABRKX3 | 602477 | ABRKX4 |
| 602500 | ABRKX5 | 602501 | DAPX0 | 602502 | DAPX1 |
| 602503 | DAPX2 | 602504 | DAPX3 | 602505 | DAPX4 |
| 602506 | DAPX5 | 602507 | DAPX6 | 602510 | DAPX7 |
| 602511 | DAPX8 | 602512 | DAPX9 | 602513 | DAPX10 |
| 602514 | DAPX11 | 602515 | DAPX12 | 602516 | DAPX13 |
| 602517 | DAPX14 | 602520 | DAPX15 | 602521 | DAPX16 |
| 602522 | DAPX17 | 602523 | DAPX18 | 602524 | DAPX19 |
| 602525 | DAPX20 | 602526 | DAPX21 | 602527 | DAPX22 |
| 602530 | DAPX23 | 602531 | DAPX24 | 602532 | DAPX25 |
| 602533 | DAPX26 | 602534 | DAPX27 | 602535 | DAPX28 |
| 602536 | DAPX29 | 602537 | DAPX30 | 602540 | CRDI25 |
| 602541 | CRDI26 | 602542 | CRDI27 | 602543 | TTYX03 |
| 602544 | CRDI28 | 602545 | NSPX27 | 602546 | GJFX55 |
| 602547 | KLPX1 | 602550 | KLPX2 | 602551 | KLPX3 |
| 602552 | KLPX4 | 602553 | MONX07 | 602554 | DCNX16 |
| 602555 | NSJX01 | 602556 | NSJX02 | 602557 | NSJX03 |
| 602560 | NSJX04 | 602561 | NSJX05 | 602562 | NSJX06 |
| 602563 | NSJX07 | 602564 | NSJX08 | 602565 | NSJX09 |
| 602566 | SCSBFC | 602567 | SCSBTS | 602570 | SCSIAB |
| 602571 | SCSAAB | 602572 | SCSNSN | 602573 | SCSNEP |
| 602574 | SCSNSC | 602575 | SCSDCB | 602576 | NODX04 |
| 602577 | NODX05 | 602600 | NODX06 | 602601 | SCSNRT |
| 602602 | SCAPTL | 602603 | SCSIID | 602604 | SCSNPA |
| 602605 | SCSNBA | 602606 | SCSZLP | 602607 | SCSSCP |
| 602610 | SCSNSD | 602611 | SCSDTL | 602612 | SCSUPC |
| 602613 | SCSQIE | 602614 | DIAG13 | 602615 | MSTX45 |
| 602616 | MSTX46 | 602617 | SCSFRK | 602620 | SCSNMQ |
| 602621 | SCSISB | 602622 | SCSNSH | 602623 | SCSIAA |
| 602624 | SCSIBP | 602645 | SCSNDQ | 602646 | SCSJBD |
| 602647 | NODX07 | 602650 | NODX10 | 602651 | NODX11 |
| 602652 | SCLX01 | 602653 | SCLX02 | 602654 | SCLX03 |
| 602655 | SCLX04 | 602656 | SCLX05 | 602657 | SCLX06 |
| 602660 | SCLX07 | 602661 | SCLX08 | 602662 | SCLX09 |
| 602663 | SCLX10 | 602664 | SCLX11 | 602665 | SCLX12 |
| 602666 | SCLX13 | 602667 | SCLX14 | 602670 | SCLX15 |
| 602671 | SCLX16 | 602672 | SCLX17 | 602673 | SCLX18 |
| 602674 | SCLX19 | 602675 | SCLX20 | 602676 | SCLX21 |
| 602677 | SCLX22 | 602700 | NODX12 | 602701 | NODX13 |
| 602702 | NODX14 | 602703 | NODX15 | 602704 | SCSENB |
| 602705 | DIAG14 | 602706 | DIAG15 | 602707 | DIAG16 |
| 602710 | SCSSTL | 602711 | SCSTMS | 602712 | DIAG17 |
| 602713 | DIAG20 | 602714 | SCSCWS | 602715 | SCSNEC |
| 602716 | SCSBAS | 602717 | SCSNSB | 602720 | SCSNEB |
| 602721 | SCSNKP | 602722 | SCSIPC | 602723 | SCSIPS |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 602724 | SCSIFL | 602725 | SCSIST | 602726 | SCSIDM |
| 602727 | KLPX5 | 602730 | KLPX6 | 602731 | KLPX7 |
| 602732 | KLPX8 | 602733 | KLPX9 | 602734 | KLPX10 |
| 602735 | KLPX11 | 602736 | CFGBFC | 602737 | CFGBTS |
| 602740 | CFGIAB | 602741 | CFGAAB | 602742 | CFGINA |
| 602743 | TTMSX2 | 602744 | XPEK01 | 602745 | XPEK02 |
| 602746 | KLPX12 | 602747 | XPEK03 | 602750 | XPEK04 |
| 602751 | NTMX2 | 602752 | KLPX13 | 602753 | MTOX21 |
| 602754 | KLPX14 | 602755 | KLPX15 | 602756 | NODX16 |
| 602757 | DKOP01 | 602760 | DKOP02 | 602761 | DKOP03 |
| 602762 | DKOP04 | 602763 | DKOP05 | 602764 | SCSIBN |
| 602765 | NTMX3 | 602766 | NODX17 | 602767 | DIAG21 |
| 602770 | DKOP06 | 602771 | DKOP07 | 602772 | CRDI29 |
| 602773 | ENQX24 | 603033 | MSTX43 | 603400 | TCPXX1 |
| 603401 | TCPXX2 | 603402 | TCPXX3 | 603403 | TCPXX4 |
| 603404 | TCPXX5 | 603405 | TCPXX6 | 603406 | TCPXX7 |
| 603407 | TCPXX8 | 603410 | TCPXX9 | 603411 | TCPX10 |
| 603412 | TCPX11 | 603413 | TCPX12 | 603414 | TCPX13 |
| 603415 | TCPX14 | 603416 | TCPX15 | 603417 | TCPX16 |
| 603420 | TCPX17 | 603421 | TCPX18 | 603422 | TCPX19 |
| 603423 | TCPX20 | 603424 | TCPX21 | 603425 | TCPX22 |
| 603426 | TCPX23 | 603427 | TCPX24 | 603430 | TCPX25 |
| 603431 | TCPX26 | 603432 | TCPX27 | 603433 | TCPX28 |
| 603434 | TCPX29 | 603435 | TCPX30 | 603436 | TCPX31 |
| 603437 | TCPX32 | 603440 | TCPX33 | 603441 | TCPX34 |
| 603442 | TCPX35 | 603443 | TCPX36 | 603444 | TCPX37 |
| 603445 | TCPX40 | 603446 | TCPX41 | 603447 | TCPX42 |
| 603450 | TCPX43 | 603451 | IPHCHK | 603452 | IPHCNT |
| 603453 | IPHNSP | 603454 | IPHEMX | 603455 | IPHSEQ |
| 603456 | IPFLAD | 603457 | ARPNSP | 603460 | IPARP1 |
| 603461 | TCPX44 | 604000 | LLMX01 | 604001 | LLMX02 |
| 604002 | LLMX03 | 604003 | LLMX04 | 604004 | LLMX05 |
| 604005 | LLMX06 | 604777 | LLMX99 | 605000 | IPCF36 |
| 605001 | MSTX44 | 605010 | LATX01 | 605011 | LATX02 |
| 605012 | LATX03 | 605013 | LATX04 | 605014 | LATX05 |
| 605015 | LATX06 | 605016 | LATX07 | 605017 | LATX08 |
| 605020 | LATX09 | 605021 | LATX10 | 605022 | LATX11 |
| 605403 | NIENSC | 605405 | NIEIVP | 605406 | NIEPIU |
| 605407 | NIEPRA | 605411 | NIENSP | 605412 | NIEIFB |
| 605413 | NIEIBS | 605414 | NIERDL | 605415 | NIERAB |
| 605416 | NIELER | 605417 | NIENPE | 605420 | NIEIBP |
| 605421 | NIEEXC | 605422 | NIEDNS | 605423 | NIENRE |
| 605424 | NIEANE | 605425 | NIEIMA | 605426 | NIEICA |
| 605427 | NIEPWS | 605431 | NIECCF | 605432 | NIESHT |
| 605433 | NIEOPN | 605434 | NIERFD | 605435 | NIEICS |
| 605436 | NIECAB | 605500 | NIERTE | 605501 | NIECIO |
| 605502 | MSCPX4 | 605600 | ARGX32 | 605601 | GNJFX2 |
| 605602 | TTYX04 | 605603 | COMX22 | 605604 | COMX23 |
| 605605 | TTMSX3 | 605606 | INFX01 | 605607 | INFX02 |
| 605610 | INFX03 | 605611 | INFX04 | 605612 | INFX05 |
| 605613 | INFX06 | 605614 | INFX07 | 605615 | INFX08 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 605616 | INFX09 | 605617 | INFX10 | 605620 | INFX11 |
| 605621 | INFX12 | 605622 | INFX13 | 605623 | INFX14 |
| 605624 | INFX15 | 605625 | INFX16 | 605626 | TTMSX4 |
| 605627 | INFX17 | 605630 | SMONX4 | 605631 | CRDI30 |
| 605634 | SMONX5 | 605635 | SMONX6 | 605637 | CRDI31 |
| 605640 | CRDI32 | 605641 | CRDI33 | | |

| Mnemonic | Code | Text String |
|----------|--------|---|
| ABRKX1 | 602123 | Address break not available on this system |
| ABRKX2 | 602475 | Address break facility is in use for system debugging |
| ABRKX3 | 602476 | Use .ABRRG function to read break conditions |
| ABRKX4 | 602477 | AB%SEC is invalid on this processor |
| ABRKX5 | 602500 | Lower and upper bounds must be equal on this processor |
| ACESX1 | 601341 | Argument block too small |
| ACESX2 | 601342 | Insufficient system resources |
| ACESX3 | 601431 | Password is required |
| ACESX4 | 601432 | Function not allowed for another job |
| ACESX5 | 601433 | No function specified for ACCES |
| ACESX6 | 601435 | Directory is not accessed |
| ACESX7 | 602137 | Directory is "files-only" and cannot be accessed |
| ALCX1 | 601137 | Invalid function |
| ALCX2 | 601140 | WHEEL or OPERATOR capability required |
| ALCX3 | 601141 | Device is not assignable |
| ALCX4 | 601142 | Invalid job number |
| ALCX5 | 601143 | Device already assigned to another job |
| ALCX6 | 601153 | Device assigned to user job, but will be given to allocator when released |
| ANTX01 | 602454 | No more network terminals available |
| ARCFX2 | 602312 | File already has archive status |
| ARCFX3 | 602313 | Cannot perform ARCF functions on nonmultiple directory devices |
| ARCFX4 | 602314 | File is not on line |
| ARCFX5 | 602315 | Files not on the same device or structure |
| ARCFX6 | 602316 | File does not have archive status |
| ARCFX7 | 602317 | Invalid parameter |
| ARCFX8 | 602320 | Archive not complete |
| ARCFX9 | 602321 | File not off line |
| ARCX10 | 602322 | Archive prohibited |
| ARCX11 | 602323 | Archive requested, modification prohibited |
| ARCX12 | 602324 | Archive requested, delete prohibited |
| ARCX13 | 602325 | Archive system request not completed |
| ARCX14 | 602332 | File restore failed |
| ARCX15 | 602333 | Migration prohibited |
| ARCX16 | 602334 | Cannot exempt off-line file |
| ARCX17 | 602335 | FDB incorrect format for ARCF JSYS |
| ARCX18 | 602336 | Retrieval request cannot be fulfilled for waiting process |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| ARCX19 | 602337 | Migration already pending |
| ARGX01 | 601705 | Invalid password |
| ARGX02 | 601713 | Invalid function |
| ARGX03 | 601714 | Illegal to change specified bits |
| ARGX04 | 601715 | Argument block too small |
| ARGX05 | 601716 | Argument block too long |
| ARGX06 | 601717 | Invalid page number |
| ARGX07 | 601720 | Invalid job number |
| ARGX08 | 601721 | No such job |
| ARGX09 | 601722 | Invalid byte size |
| ARGX10 | 601723 | Invalid access requested |
| ARGX11 | 601724 | Invalid directory number |
| ARGX12 | 601725 | Invalid process handle |
| ARGX13 | 601726 | Invalid software interrupt channel number |
| ARGX14 | 601733 | Invalid account identifier |
| ARGX15 | 601734 | Job is not logged in |
| ARGX16 | 601741 | Password is required |
| ARGX17 | 601742 | Invalid argument block length |
| ARGX18 | 601743 | Invalid structure name |
| ARGX19 | 602033 | Invalid unit number |
| ARGX20 | 602171 | Invalid arithmetic trap argument |
| ARGX21 | 602172 | Invalid LUUO trap argument |
| ARGX22 | 602173 | Invalid flags |
| ARGX23 | 602177 | Invalid section number |
| ARGX24 | 602200 | Invalid count |
| ARGX25 | 602246 | Invalid class |
| ARGX26 | 602340 | File is off line |
| ARGX27 | 602341 | Off line expiration time cannot exceed system or directory maximum |
| ARGX28 | 602412 | not available on this system |
| ARGX29 | 602414 | Invalid class share |
| ARGX30 | 602415 | Invalid KNOB value |
| ARGX31 | 602416 | Class Scheduler already enabled |
| ARGX32 | 605600 | On line expiration cannot exceed system or directory maximum |
| ARPNSP | 603457 | Insufficient system resources (No space for ARP buffers) |
| ASNDX1 | 600300 | Device is not assignable |
| ASNDX2 | 600301 | Illegal to assign this device |
| ASNDX3 | 600302 | No such device |
| ASNSX1 | 600740 | Insufficient system resources (All special queues in use) |
| ASNSX2 | 600741 | Link(s) assigned to another special queue |
| ATACX1 | 600320 | Invalid job number |
| ATACX2 | 600321 | Job already attached |
| ATACX3 | 600322 | Incorrect user number |
| ATACX4 | 600323 | Invalid password |
| ATACX5 | 600324 | This job has no controlling terminal |
| ATACX6 | 601502 | Terminal is already attached to a job |
| ATACX7 | 601503 | Illegal terminal number |
| ATIX1 | 600352 | Invalid software interrupt channel number |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| ATIX2 | 600353 | Control-C capability required |
| ATNX1 | 600710 | Invalid receive JFN |
| ATNX10 | 600721 | Send JFN is not a NET connection |
| ATNX11 | 600722 | Send JFN has been used |
| ATNX12 | 600723 | Send connection refused |
| ATNX13 | 600724 | Insufficient system resources (No NVT's) |
| ATNX2 | 600711 | Receive JFN not opened for read |
| ATNX3 | 600712 | Receive JFN not open |
| ATNX4 | 600713 | Receive JFN is not a NET connection |
| ATNX5 | 600714 | Receive JFN has been used |
| ATNX6 | 600715 | Receive connection refused |
| ATNX7 | 600716 | Invalid send JFN |
| ATNX8 | 600717 | Send JFN not opened for write |
| ATNX9 | 600720 | Send JFN not open |
| ATSX01 | 602146 | Invalid mode |
| ATSX02 | 602147 | Illegal to declare mode twice |
| ATSX03 | 602150 | Illegal to declare mode after acquiring terminal |
| ATSX04 | 602151 | Invalid event code |
| ATSX05 | 602152 | Invalid function code for channel assignment |
| ATSX06 | 602153 | JFN is not an ATS JFN |
| ATSX07 | 602154 | Table length too small |
| ATSX08 | 602155 | Table lengths must be the same |
| ATSX09 | 602156 | Table length too large |
| ATSX10 | 602157 | Maximum applications terminals for system already assigned |
| ATSX11 | 602160 | Byte count is too large |
| ATSX12 | 602161 | Terminal not assigned to this JFN |
| ATSX13 | 602162 | Terminal is XOFF'd |
| ATSX14 | 602163 | Terminal has been released |
| ATSX15 | 602164 | Terminal identifier is not assigned |
| ATSX16 | 602166 | Invalid Host Terminal Number |
| ATSX17 | 602167 | Output failed -- monitor internal error |
| ATSX18 | 602174 | ATS input message too long for internal buffers |
| ATSX19 | 602175 | Monitor internal error - ATS input message truncated |
| ATSX20 | 602176 | Illegal to close JFN with terminal assigned |
| ATSX21 | 602213 | Maximum applications terminals for job already assigned |
| ATSX22 | 602214 | Failed to acquire applications terminal |
| ATSX23 | 602215 | Invalid device name |
| ATSX24 | 602216 | Invalid server name |
| ATSX25 | 602217 | Terminal is already released |
| ATSX26 | 602226 | Invalid host name |
| ATSX27 | 602276 | Terminal is not open |
| ATSX28 | 602277 | Unknown error received |
| ATSX29 | 602300 | Receive error threshold exceeded |
| ATSX30 | 602301 | Reply threshold exceeded |
| ATSX31 | 602302 | NAK threshold exceeded |
| ATSX32 | 602303 | Terminal protocol error |
| ATSX33 | 602304 | Intervention required at terminal |
| ATSX34 | 602305 | Powerfail |
| ATSX35 | 602306 | Data pipe was disconnected |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|---------|--------|--|
| AT SX36 | 602307 | Dialup terminal was attached |
| BKJFX1 | 600454 | Illegal to back up terminal pointer twice |
| BOTX01 | 602016 | Invalid DTE-20 number |
| BOTX02 | 602017 | Invalid byte size |
| BOTX03 | 602031 | Invalid protocol version number |
| BOTX04 | 602114 | Byte count is not positive |
| BOTX05 | 602132 | Protocol initialization failed |
| BOTX06 | 602434 | GTJFN failed for dump file |
| BOTX07 | 602435 | OPENF failed for dump file |
| BOTX08 | 602436 | Dump failed |
| BOTX09 | 602437 | To -10 error on dump |
| BOTX10 | 602440 | To -11 error on dump |
| BOTX11 | 602441 | Failed to assign page on dump |
| BOTX12 | 602442 | Reload failed |
| BOTX13 | 602443 | -11 didn't power down |
| BOTX14 | 602444 | -11 didn't power up |
| BOTX15 | 602445 | ROM did not ACK the -10 |
| BOTX16 | 602446 | -11 boot program did not make it to -11 |
| BOTX17 | 602447 | -11 took more than 1 minute to reload. Will cause retry |
| BOTX18 | 602450 | Unknown BOOT error |
| BOTX19 | 602467 | Overdue To-11 transfer aborted |
| BOTX20 | 602470 | Overdue To-10 transfer aborted |
| CACTX1 | 600045 | Invalid account identifier |
| CACTX2 | 600046 | Job is not logged in |
| CAPX1 | 600615 | WHEEL or OPERATOR capability required |
| CAPX2 | 601231 | WHEEL, OPERATOR, or MAINTENANCE capability required |
| CAPX3 | 601706 | WHEEL capability required |
| CAPX4 | 601707 | WHEEL or IPCF capability required |
| CAPX6 | 601711 | ENQ/DEQ capability required |
| CAPX7 | 601712 | Confidential Information Access Capability required |
| CFDBX1 | 600430 | Invalid displacement |
| CFDBX2 | 600431 | Illegal to change specified bits |
| CFDBX3 | 600432 | Write or owner access required |
| CFDBX4 | 600433 | Invalid value for specified bits |
| CFDBX5 | 600434 | No FDB for non-directory devices |
| CFGAAB | 602741 | Error accessing argument block |
| CFGBFC | 602736 | Function code out of range |
| CFGBTS | 602737 | Argument block too short |
| CFG IAB | 602740 | Invalid argument block address |
| CFG INA | 602742 | Information not available for this function |
| CFR KX3 | 600363 | Insufficient system resources |
| CIBDCD | 601471 | Bad CI op code |
| CIBDFQ | 601501 | BAD CI FREE QUEUE |
| CIBDOF | 601465 | BAD BDT offset given |
| CILNER | 601474 | CI length error |
| CIMXND | 601463 | Maximum memory driver nodes assigned |
| CINOFQ | 601466 | No CI free queue entries left |
| CINOND | 601464 | No LCS node slots availble |
| CINOND | 601473 | Dead LCS node |
| CINOPG | 601467 | No BDT page slots left |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| CINPTH | 601470 | Target CI LCS node is dead, no path to it |
| CIUNOP | 601472 | Undefined op code (in range but not yet defined) |
| CKAX1 | 601154 | Argument block too small |
| CKAX2 | 601155 | Invalid directory number |
| CKAX3 | 601156 | Invalid access code |
| CKAX4 | 601271 | File is not on disk |
| CLSX1 | 600160 | File is not open |
| CLSX2 | 600161 | File cannot be closed by this process |
| CLSX3 | 601151 | File still mapped |
| CLSX4 | 601217 | Device still active |
| CNDIX1 | 600200 | Invalid password |
| CNDIX2 | 601241 | WHEEL or OPERATOR capability required |
| CNDIX3 | 600202 | Invalid directory number |
| CNDIX4 | 601242 | Invalid job number |
| CNDIX5 | 600204 | Job is not logged in |
| CNDIX6 | 601243 | Job is not logged in |
| CNDIX7 | 602004 | The CNDIR JSYS has been replaced by ACCES |
| COMNX1 | 601257 | Invalid COMND function code |
| COMNX2 | 601260 | Field too long for internal buffer |
| COMNX3 | 601261 | Command too long for internal buffer |
| COMNX4 | 601262 | Invalid character in input |
| COMNX5 | 601265 | Invalid string pointer argument |
| COMNX6 | 601266 | Problem in indirect file |
| COMNX7 | 601267 | Error in command |
| COMNX8 | 601321 | Number base out of range 2-10 |
| COMNX9 | 601413 | End of input file reached |
| COMX10 | 601767 | Invalid default string |
| COMX11 | 602035 | Invalid CMRTY pointer |
| COMX12 | 602036 | Invalid CMBFP pointer |
| COMX13 | 602037 | Invalid CMPTR pointer |
| COMX14 | 602040 | Invalid CMABP pointer |
| COMX15 | 602041 | Invalid default string pointer |
| COMX16 | 602042 | Invalid help message pointer |
| COMX17 | 602043 | Invalid byte pointer in function block |
| COMX18 | 602134 | Invalid character in node name |
| COMX19 | 602135 | Too many characters in node name |
| COMX20 | 602212 | Invalid node name |
| COMX21 | 602452 | Node name doesn't contain an alphabetic character |
| COMX22 | 605603 | Invalid use of quoting character in directory name |
| COMX23 | 605604 | Invalid use of quoting character in username |
| CRDI10 | 601170 | Maximum directory number exceeded; index table needs expanding |
| CRDI11 | 601427 | Invalid terminating bracket on directory |
| CRDI12 | 601451 | Structure is not mounted |
| CRDI13 | 602101 | Request exceeds superior directory working quota |
| CRDI14 | 602102 | Request exceeds superior directory permanent quota |
| CRDI15 | 602103 | Request exceeds superior directory subdirectory quota |
| CRDI16 | 602104 | Invalid user group |
| CRDI17 | 602117 | Illegal to create non-files-only subdirectory under files-only directory |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|---------|--------|---|
| CRDI18 | 602127 | Illegal to delete logged-in directory |
| CRDI19 | 602130 | Illegal to delete connected directory |
| CRDI20 | 602133 | WHEEL, OPERATOR, or requested capability required |
| CRDI21 | 602136 | Working space insufficient for current allocation |
| CRDI22 | 602140 | Subdirectory quota insufficient for existing subdirectories |
| CRDI23 | 602141 | Superior directory does not exist |
| CRDI24 | 602144 | Invalid subdirectory quota |
| CRDI25 | 602540 | Maximum number of remote aliases exceeded |
| CRDI26 | 602541 | CRDIR block does not include password encryption version |
| CRDI27 | 602542 | Attempt to use encrypted password on unencrypted structure |
| CRDI28 | 602544 | Invalid password encryption version number |
| CRDI29 | 602772 | Illegal to disallow subdirectory user group while in use |
| CRDI30 | 605631 | Insufficient password length |
| CRDI31 | 605637 | Password expiration date is too far in the future |
| CRDI32 | 605640 | Password expiration is not enabled on this system |
| CRDI33 | 605641 | Password found in system password dictionary |
| CRDIX1 | 600620 | WHEEL or OPERATOR capability required |
| CRDIX2 | 600621 | Illegal to change number of old directory |
| CRDIX3 | 600622 | Insufficient system resources (Job Storage Block full) |
| CRDIX4 | 600623 | Superior directory full |
| CRDIX5 | 600624 | Directory name not given |
| CRDIX6 | 601412 | Directory file is mapped |
| CRDIX7 | 600626 | File(s) open in directory |
| CRDIX8 | 601166 | Invalid directory number |
| CRDIX9 | 601167 | Internal format of directory is incorrect |
| CRJ BX1 | 600020 | Invalid parameter or function bit combination |
| CRJ BX2 | 600021 | Illegal for created job to enter MINI-EXEC |
| CRJ BX3 | 600022 | Reserved |
| CRJ BX4 | 600023 | Terminal is not available |
| CRJ BX5 | 600024 | Unknown name for LOGIN |
| CRJ BX6 | 600025 | Insufficient system resources |
| CRJ BX7 | 600026 | Reserved |
| CRLNX1 | 601000 | Logical name is not defined |
| CRLNX2 | 601134 | WHEEL or OPERATOR capability required |
| CRLNX3 | 601152 | Invalid function |
| CTS X01 | 601600 | CTSOP% Function Code Out of Range |
| CTS X02 | 601601 | Undefined CTSOP% Function |
| CTS X03 | 601602 | Insufficient System Resources (No JSB Free Space) |
| CTS X04 | 601603 | No Default Canonical Library Name |
| CTS X05 | 601604 | Illegal to Issue CTSOP% .CTCAL Function from Section Zero |
| CTS X06 | 601605 | Stack Overflow During CTSOP% .CTCAL Function |
| CTS X07 | 601606 | Illegal Memory Write During CTSOP% .CTCAL Function |
| CTS X08 | 601607 | Invalid Function Code Given During CTSOP% .CTCAL Function |
| CTS X09 | 601610 | No Address of CTS Descriptor Block Found in Library |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|---------|--|
| | | Descriptor Block of Library |
| CTSX10 | 601611 | Length of CTS Descriptor Block Incorrect |
| CTSX11 | 601612 | Invalid Number of Pages in CTS Descriptor Block |
| CTSX12 | 601613 | No Monitor Pages Available for Terminal Data Base |
| CTSX13 | 601614 | Unimplemented Canonical Terminal Operation |
| CVHST1 | 600727 | No string for that Host number |
| CVSKX1 | 600730 | Invalid network JFN |
| CVSKX2 | 600731 | Local socket invalid in this context |
| DAPX0 | 602501 | Illegal DAP% function code |
| DAPX1 | 602502 | Nested ACLREPs in formatting table not allowed |
| DAPX10 | 602513 | LENGTH or LEN256 field present in message block |
| DAPX11 | 602514 | Protocol error on receive, DAP length exceeds DECnet length |
| DAPX12 | 602515 | Message type is not DATA, yet there is a BITCNT field |
| DAPX13 | 602516 | Field following ACLREP is not VALUE1 or VALUE2 |
| DAPX14 | 602517 | Invalid link handle |
| DAPX15 | 602520 | Transmission in progress, AC2 has retry message block addr |
| DAPX16 | 602521 | CONTINUE TRANSFER message cannot be sent as normal message |
| DAPX17 | 602522 | Only CONTINUE TRANSFER messages can be sent as interrupt |
| DAPX18 | 602523 | Interrupt messages cannot be sent blocked |
| DAPX19 | 602524 | There is already an interrupt transmission in progress |
| DAPX2 | 602503 | Parse error, fixed length field has wrong length |
| DAPX20 | 602525 | Receive in progress |
| DAPX21 | 602526 | There is no interrupt message available |
| DAPX22 | 602527 | Illegal function for passive link |
| DAPX23 | 602530 | Illegal function for active link |
| DAPX24 | 602531 | There is no message available |
| DAPX25 | 602532 | Protocol error on receive, message was too long |
| DAPX2 | 6602533 | Too many message blocks chained together |
| DAPX27 | 602534 | Illegal function for this state |
| DAPX28 | 602535 | Feature not supported by remote server |
| DAPX29 | 602536 | Protocol error on receive - wrong message type |
| DAPX3 | 602504 | Parse error, expecting more bytes |
| DAPX30 | 602537 | No alias for this node |
| DAPX4 | 602505 | Message byte length was too long for this link |
| DAPX5 | 602506 | Parse error, variable length field was too long |
| DAPX6 | 602507 | Parse error, bit mask was too long |
| DAPX7 | 602510 | Illegal DAP% message type |
| DAPX8 | 602511 | Protocol error on receive, LEN256 field without LENGTH field |
| DAPX9 | 602512 | Parse error on receive, extra bytes at end of message |
| DATEX1 | 600466 | Year out of range |
| DATEX2 | 600467 | Month is not less than 12 |
| DATEX3 | 600470 | Day of month too large |
| DATEX4 | 600471 | Day of week is not less than 7 |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|---------|--------|---|
| DATEX5 | 600472 | Date out of range |
| DATEX6 | 600473 | System date and time are not set |
| DATEX7 | 602310 | Julian day is out of range |
| DBRKX1 | 601275 | No interrupts in progress |
| DCNX1 | 602020 | Invalid network file name |
| DCNX11 | 602026 | Link aborted |
| DCNX12 | 602027 | String exceeds 16 bytes |
| DCNX13 | 602202 | Node not accessible |
| DCNX14 | 602203 | Previous interrupt message outstanding |
| DCNX15 | 602204 | No interrupt message available |
| DCNX16 | 602554 | Illegal operation for current link state |
| DCNX2 | 602122 | Interrupt message must be read first |
| DCNX3 | 602022 | Invalid object |
| DCNX4 | 602023 | Invalid task name |
| DCNX5 | 602021 | No more logical links available |
| DCNX8 | 602025 | Invalid network operation |
| DCNX9 | 602024 | Object is already defined |
| DECRSV | 601456 | DEC reserved bits not zero |
| DELDX1 | 601171 | WHEEL or OPERATOR capability required |
| DELDX2 | 601172 | Invalid directory number |
| DELFX10 | 602100 | Directory still contains subdirectory |
| DELFX1 | 600170 | Delete access required |
| DELFX2 | 601303 | File cannot be expunged because it is currently open |
| DELFX3 | 601304 | System scratch area depleted; file not deleted |
| DELFX4 | 601305 | Directory symbol table could not be rebuilt |
| DELFX5 | 601306 | Directory symbol table needs rebuilding |
| DELFX6 | 601307 | Internal format of directory is incorrect |
| DELFX7 | 601310 | FDB formatted incorrectly; file not deleted |
| DELFX8 | 601311 | FDB not found; file not deleted |
| DELFX9 | 601411 | File is not a directory file |
| DELX11 | 602330 | File has archive status, delete is not permitted |
| DELX12 | 602331 | File has no pointer to offline storage |
| DELX13 | 602453 | File is marked "Never Delete" |
| DESX1 | 600150 | Invalid source/destination designator |
| DESX10 | 601417 | Structure is dismounted |
| DESX11 | 602410 | Invalid operation for this label type |
| DESX2 | 600151 | Terminal is not available to this job |
| DESX3 | 600152 | JFN is not assigned |
| DESX4 | 600153 | Invalid use of terminal designator or string pointer |
| DESX5 | 600154 | File is not open |
| DESX6 | 600155 | Device is not a terminal |
| DESX7 | 600156 | Illegal use of parse-only JFN or output wildcard-designators |
| DESX8 | 600157 | File is not on disk |
| DESX9 | 601340 | Invalid operation for this device |
| DEVX1 | 600335 | Invalid device designator |
| DEVX2 | 600336 | Device already assigned to another job |
| DEVX3 | 600337 | Device is not on line |
| DEVX4 | 601737 | Device is not assignable |
| DEVX5 | 601744 | No such device |
| DEVX6 | 602275 | Job has open JFN on device |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| DEVX7 | 602417 | Null device name given |
| DIAG10 | 601205 | Subunit does not exist |
| DIAG11 | 602406 | Unit already online |
| DIAG12 | 602407 | Unit not online |
| DIAG13 | 602614 | Datagram buffer not available |
| DIAG14 | 602705 | Port doesn't exist or is not a CI port |
| DIAG15 | 602706 | CI counters not available |
| DIAG16 | 602707 | Fork doesn't own CI counters |
| DIAG17 | 602712 | CI chan is not enabled |
| DIAG20 | 602713 | Diagnostic owns the channel |
| DIAG21 | 601513 | Performance counter read timed out |
| DIAG21 | 602767 | DIAG% Illegal for Dual Ported Disks |
| DIAG22 | 601515 | Illegal CI node number |
| DIAG23 | 601516 | No System Block for Remote CI Node |
| DIAG24 | 601517 | Remote CI Node does not support this function |
| DIAG25 | 601520 | Remote CI Node not in correct state for this function |
| DIAG26 | 601521 | Illegal argument for this DIAG% function |
| DIAG27 | 601522 | Read/Write of CI Maintenance data timed out |
| DIAG30 | 601523 | Read/Write of CI Maintenance data finished with an error |
| DIAGX1 | 601174 | Invalid function |
| DIAGX2 | 601175 | Device is not assigned |
| DIAGX3 | 601176 | Argument block too small |
| DIAGX4 | 601177 | Invalid device type |
| DIAGX5 | 601200 | WHEEL, OPERATOR, or MAINTENANCE capability required |
| DIAGX6 | 601201 | Invalid channel command list |
| DIAGX7 | 601202 | Illegal to do I/O across page boundary |
| DIAGX8 | 601203 | No such device |
| DIAGX9 | 601204 | Unit does not exist |
| DILFX1 | 600464 | Invalid date format |
| DIRX1 | 601313 | Invalid directory number |
| DIRX2 | 601314 | Insufficient system resources |
| DIRX3 | 601315 | Internal format of directory is incorrect |
| DIRX4 | 601745 | Invalid directory specification |
| DIRX5 | 602342 | Directory too large |
| DKOP01 | 602757 | Illegal disk address |
| DKOP02 | 602760 | Transfer too large |
| DKOP03 | 602761 | Invalid unit specified |
| DKOP04 | 602762 | Illegal address specified |
| DKOP05 | 602763 | Size not sector size |
| DKOP06 | 602770 | Data or device error |
| DKOP07 | 602771 | Device is offline |
| DLFX10 | 602010 | Cannot delete directory; file still mapped |
| DLFX11 | 602011 | Cannot delete directory file in this manner |
| DOBX01 | 601615 | Not a BUGCHK or BUGINF |
| DOBX02 | 601616 | DOB is disabled |
| DOBX03 | 601617 | DOB already disabled |
| DOBX04 | 601620 | DOB already enabled |
| DOBX05 | 601621 | Dump was not requested for this BUG |
| DOBX06 | 601622 | Dump was already requested for this BUG |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| DOBX07 | 601623 | Structure is not dumpable |
| DOBX08 | 601624 | DOB timeout out of range |
| DSKOX1 | 601343 | Channel number too large |
| DSKOX2 | 601344 | Unit number too large |
| DSKOX3 | 601416 | Invalid structure number |
| DSKOX4 | 601420 | Invalid address type specified |
| DSKOX5 | 601533 | Invalid word count |
| DSKOX6 | 601534 | Invalid buffer address |
| DSKX01 | 601365 | Invalid structure number |
| DSKX02 | 601366 | Bit table is being initialized |
| DSKX03 | 601367 | Bit table has not been initialized |
| DSKX04 | 601370 | Bit table being initialized by another job |
| DSKX05 | 601763 | Disk assignments and deassignments are currently prohibited |
| DSKX06 | 601764 | Invalid disk address |
| DSKX07 | 601765 | Address cannot be deassigned because it is not assigned |
| DSKX08 | 601766 | Address cannot be assigned because it is already assigned |
| DSMX1 | 600555 | File(s) not closed |
| DUMPX1 | 600440 | Command list error |
| DUMPX2 | 600441 | JFN is not open in dump mode |
| DUMPX3 | 600442 | Address error (too big or crosses end of memory) |
| DUMPX4 | 600443 | Access error (cannot read or write data in memory) |
| DUMPX5 | 601214 | No-wait dump mode not supported for this device |
| DUMPX6 | 601215 | Dump mode not supported for this device |
| DYNX01 | 601561 | DYNLB% Function Code Out of Range |
| DYNX02 | 601562 | Undefined DYNLB% Function |
| DYNX03 | 601563 | No Free Section In Which to Map Dynamic Library |
| DYNX04 | 601564 | Unable to Get a JFN on Dynamic Library File |
| DYNX05 | 601565 | Unable to Get Dynamic Library |
| DYNX06 | 601566 | No Program Data Vector Found in Dynamic Library |
| DYNX07 | 601567 | More Than One Dynamic Library in File |
| DYNX08 | 601570 | Unable to Un-Map Section During De-Link Operation |
| DYNX09 | 601571 | No Transfer Vector Address in Library Descriptor Block of Dynamic Library |
| DYNX10 | 601572 | Library Name String Too Long |
| DYNX11 | 601573 | Unable to Make Library Known (No JSB Free Space) |
| EFCTX1 | 600050 | WHEEL or OPERATOR capability required |
| EFCTX2 | 600051 | Entry cannot be longer than 64 words |
| EFCTX3 | 600052 | Fatal error when accessing FACT file |
| ENACX1 | 602105 | Account validation data base file not completely closed |
| ENACX2 | 602106 | Cannot get a JFN for <SYSTEM>ACCOUNTS-TABLE.BIN |
| ENACX3 | 602107 | Account validation data base file too long |
| ENACX4 | 602110 | Cannot get an OFN for <SYSTEM>ACCOUNTS-TABLE.BIN |
| ENACX5 | 602131 | Account validation data base file is empty |
| ENQX1 | 601055 | Invalid function |
| ENQX10 | 601066 | Invalid argument block length |
| ENQX11 | 601067 | Invalid software interrupt channel number |
| ENQX12 | 601070 | Invalid number of resources requested |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| ENQX13 | 601071 | Indirect or indexed byte pointer not allowed |
| ENQX14 | 601072 | Invalid byte size |
| ENQX15 | 601073 | ENQ/DEQ capability required |
| ENQX16 | 601074 | WHEEL or OPERATOR capability required |
| ENQX17 | 601075 | Invalid JFN |
| ENQX18 | 601076 | Quota exceeded |
| ENQX19 | 601077 | String too long |
| ENQX2 | 601056 | Level number too small |
| ENQX20 | 601100 | Locked JFN cannot be closed |
| ENQX21 | 601101 | Job is not logged in |
| ENQX22 | 602121 | Invalid mask block length |
| ENQX23 | 602120 | Mismatched mask block lengths |
| ENQX24 | 602773 | Internal resources exhausted (No more SCA buffers) |
| ENQX3 | 601057 | Request and lock level numbers do not match |
| ENQX4 | 601060 | Number of pool and lock resources do not match |
| ENQX5 | 601061 | Lock already requested |
| ENQX6 | 601062 | Requested locks are not all locked |
| ENQX7 | 601063 | No ENQ on this lock |
| ENQX8 | 601064 | Invalid access change requested |
| ENQX9 | 601065 | Invalid number of blocks specified |
| FDFRX1 | 600700 | Not a multiple-directory device |
| FDFRX2 | 600701 | Invalid directory number |
| FFFFX1 | 601457 | No free pages in file |
| FFUFX1 | 600544 | File is not open |
| FFUFX2 | 600545 | File is not on multiple-directory device |
| FFUFX3 | 600546 | No used page found |
| FILX01 | 601704 | File is not open |
| FILX02 | 601735 | Write or owner access required |
| FILX03 | 601736 | List access required |
| FILX04 | 601740 | File is not on multiple-directory device |
| FILX05 | 601746 | File expunged |
| FLINX1 | 600650 | First character is not blank or numeric |
| FLINX2 | 600651 | Number too small |
| FLINX3 | 600652 | Number too large |
| FLINX4 | 600653 | Invalid format |
| FLOTX1 | 600660 | Column overflow in field 1 or 2 |
| FLOTX2 | 600661 | Column overflow in field 3 |
| FLOTX3 | 600662 | Invalid format specified |
| FRKHX1 | 600250 | Invalid process handle |
| FRKHX2 | 600251 | Illegal to manipulate a superior process |
| FRKHX3 | 600252 | Invalid use of multiple process handle |
| FRKHX4 | 600253 | Process is running |
| FRKHX5 | 600254 | Process has not been started |
| FRKHX6 | 600255 | All relative process handles in use |
| FRKHX7 | 601312 | Process page cannot exceed 777 |
| FRKHX8 | 602170 | Illegal to manipulate an execute-only process |
| GACCX1 | 601272 | Invalid job number |
| GACCX2 | 601273 | No such job |
| GACCX3 | 601301 | Confidential Information Access capability required |
| GACTX1 | 600540 | File is not on multiple-directory device |
| GACTX2 | 600541 | File expunged |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| GACTX3 | 601173 | Internal format of directory is incorrect |
| GETX1 | 600373 | Invalid save file format |
| GETX2 | 600374 | System Special Pages Table full |
| GETX3 | 601703 | Illegal to overlay existing pages |
| GETX4 | 601557 | Illegal to relocate (via .GBASE) a multi-section exe file |
| GETX5 | 601560 | Exe file directory entry specifies a section-crossing |
| GFDBX1 | 600424 | Invalid displacement |
| GFDBX2 | 600425 | Invalid number of words |
| GFDBX3 | 600426 | List access required |
| GFKSX1 | 601011 | Area too small to hold process structure |
| GFRKX1 | 600371 | Invalid process handle |
| GFUSX1 | 601371 | Invalid function |
| GFUSX2 | 601372 | Insufficient system resources |
| GFUSX3 | 601452 | File expunged |
| GFUSX4 | 601453 | Internal format of directory is incorrect |
| GJFX1 | 600055 | Desired JFN invalid |
| GJFX10 | 600066 | Generation number is not numeric |
| GJFX11 | 600067 | More than one generation number field is not allowed |
| GJFX12 | 600070 | More than one account field is not allowed |
| GJFX13 | 600071 | More than one protection field is not allowed |
| GJFX14 | 600072 | Invalid protection |
| GJFX15 | 600073 | Invalid confirmation character |
| GJFX16 | 600074 | No such device |
| GJFX17 | 600075 | No such directory name |
| GJFX18 | 600076 | No such filename |
| GJFX19 | 600077 | No such file type |
| GJFX2 | 600056 | Desired JFN not available |
| GJFX20 | 600100 | No such generation number |
| GJFX21 | 600101 | File was expunged |
| GJFX22 | 600102 | Insufficient system resources (Job Storage Block full) |
| GJFX23 | 600103 | Exceeded maximum number of files per directory |
| GJFX24 | 600104 | File not found |
| GJFX27 | 600107 | File already exists (new file required) |
| GJFX28 | 600110 | Device is not on line |
| GJFX29 | 600111 | Device is not available to this job |
| GJFX3 | 600057 | No JFN available |
| GJFX30 | 600112 | Account is not numeric |
| GJFX31 | 600113 | Invalid wildcard designator |
| GJFX32 | 600114 | No files match this specification |
| GJFX33 | 600115 | Filename was not specified |
| GJFX34 | 600116 | Invalid character "?" in file specification |
| GJFX35 | 600117 | Directory access privileges required |
| GJFX36 | 600760 | Internal format of directory is incorrect |
| GJFX37 | 601133 | Input deleted |
| GJFX38 | 601164 | File not found because output-only device was specified |
| GJFX39 | 601165 | Logical name loop detected |
| GJFX4 | 600060 | Invalid character in filename |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| GJFX40 | 601225 | Undefined attribute in file specification |
| GJFX41 | 601277 | File name must not exceed 6 characters |
| GJFX42 | 601300 | File type must not exceed 3 characters |
| GJFX43 | 601754 | More than one ;T specification is not allowed |
| GJFX44 | 602012 | Account string does not match |
| GJFX45 | 602060 | Illegal to request multiple specifications for the same attribute |
| GJFX46 | 602061 | Attribute value is required |
| GJFX47 | 602062 | Attribute does not take a value |
| GJFX48 | 602064 | GTJFN input buffer is empty |
| GJFX49 | 602065 | Invalid attribute for this device |
| GJFX5 | 600061 | Field cannot be longer than 39 characters |
| GJFX50 | 602205 | Invalid argument for attribute |
| GJFX51 | 602211 | Byte count too small |
| GJFX52 | 602420 | End of tape encountered while searching for file |
| GJFX53 | 602461 | Tape label filename specification exceeds 17 characters |
| GJFX54 | 602466 | Node name not first field in filespec |
| GJFX55 | 602546 | Illegal to use node name |
| GJFX6 | 600062 | Device field not in a valid position |
| GJFX7 | 600063 | Directory field not in a valid position |
| GJFX8 | 600064 | Directory terminating delimiter is not preceded by a valid beginning delimiter |
| GJFX9 | 600065 | More than one name field is not allowed |
| GNJFX1 | 601054 | No more files in this specification |
| GNJFX2 | 605601 | Could not step to next file because current file no longer exists |
| GOKER1 | 602220 | Illegal function |
| GOKER2 | 602221 | Request denied by Access Control Facility |
| GOKER3 | 602421 | JSYS not executed within ACJ fork |
| GTABX1 | 600267 | Invalid table number |
| GTABX2 | 600270 | Invalid table index |
| GTABX3 | 600271 | GETAB capability required |
| GTDIX1 | 600640 | WHEEL or OPERATOR capability required |
| GTDIX2 | 600641 | Invalid directory number |
| GTHSX1 | 600703 | No DNS name servers configured |
| GTHSX2 | 600704 | Unknown host number |
| GTHSX3 | 600705 | Unknown host name |
| GTHSX4 | 600706 | Format error in DNS message |
| GTHSX5 | 600707 | No interface to specified network |
| GTHSX6 | | Invalid class for function |
| GTHSX7 | | Server failed to find data (non-authoritative) |
| GTHSX8 | | Data not found in namespace (authoritative) |
| GTHSX9 | | String argument is too long |
| GTHX10 | | System host tables full |
| GTJIX1 | 601013 | Invalid index |
| GTJIX2 | 601014 | Invalid terminal line number |
| GTJIX3 | 601015 | Invalid job number |
| GTJIX4 | 601254 | No such job |
| GTNCX1 | 600746 | Invalid network JFN |
| GTNCX2 | 600747 | Invalid or inactive NVT |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| HFRKX1 | 600370 | Illegal to halt self with HFORK |
| HPTX1 | 600670 | Undefined clock number |
| IFIXX1 | 600414 | Radix is not in range 2 to 36 |
| IFIXX2 | 600415 | First nonspace character is not a digit |
| IFIXX3 | 600416 | Overflow (number is equal to or greater than 235) |
| ILINS1 | 600770 | Undefined operation code |
| ILINS2 | 600771 | Undefined JSYS |
| ILINS3 | 600772 | Uuo simulation facility not available |
| ILINS4 | 601255 | Uuo simulation is disabled |
| ILINS5 | 601256 | RMS facility is not available |
| ILLX01 | 601774 | Illegal memory read |
| ILLX02 | 601775 | Illegal memory write |
| ILLX03 | 601776 | Memory data parity error |
| ILLX04 | 601777 | Reference to non-existent page |
| ILLX05 | 602471 | Illegal memory reference, section greater than 37 |
| INFX01 | 605606 | Invalid INFO% function |
| INFX02 | 605607 | Invalid CI node number |
| INFX03 | 605610 | WHEEL or OPERATOR capability required |
| INFX04 | 605611 | CI node disconnected before information was returned |
| INFX05 | 605612 | Remote node not supplying information |
| INFX06 | 605613 | Insufficient system resources - no more swappable free space |
| INFX07 | 605614 | User not logged in |
| INFX08 | 605615 | Insufficient system resources on remote node (no more free space) |
| INFX09 | 605616 | Unimplemented function on remote system |
| INFX10 | 605617 | Insufficient SCA buffers to process request |
| INFX11 | 605620 | Remote system not running CLUDGR SYSAP |
| INFX12 | 605621 | Invalid argument block |
| INFX13 | 605622 | Job not logged in |
| INFX14 | 605623 | Remote node could not execute given function |
| INFX15 | 605624 | Bad argument block length |
| INFX16 | 605625 | Insufficient credit to send request to remote system |
| INFX17 | 605627 | Remote XPEEK% can only return 512 words |
| INLNX1 | 601001 | Index is beyond end of logical name table |
| INLNX2 | 601135 | Invalid function |
| IOX1 | 600215 | File is not opened for reading |
| IOX10 | 601240 | Record is longer than user requested |
| IOX11 | 601440 | Quota exceeded |
| IOX12 | 601441 | Insufficient system resources (Swapping space full) |
| IOX13 | 602227 | Invalid segment type |
| IOX14 | 602230 | Invalid segment size |
| IOX15 | 602231 | Illegal tape format for dump mode |
| IOX16 | 602232 | Density specified does not match tape density |
| IOX17 | 602233 | Invalid tape label |
| IOX2 | 600216 | File is not opened for writing |
| IOX20 | 602234 | Illegal tape record size |
| IOX21 | 602235 | Tape HDR1 missing |
| IOX22 | 602236 | Invalid tape HDR1 sequence number |
| IOX23 | 602237 | Tape label read error |
| IOX24 | 602240 | Logical end of tape encountered |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| IOX25 | 602241 | Invalid tape format |
| IOX26 | 602243 | Tape write date has not expired |
| IOX27 | 602244 | Tape is domestic and HDR2 is missing |
| IOX3 | 600217 | File is not open for random access |
| IOX30 | 602245 | Tape has invalid access character |
| IOX31 | 602343 | Invalid record descriptor in labeled tape |
| IOX32 | 602422 | Tape position is indeterminate |
| IOX33 | 602423 | TTY input buffer full |
| IOX34 | 602462 | Disk structure completely full |
| IOX35 | 602463 | Disk structure damaged, cannot allocate space |
| IOX4 | 600220 | End of file reached |
| IOX5 | 600221 | Device or data error |
| IOX6 | 600222 | Illegal to write beyond absolute end of file |
| IOX7 | 601211 | Insufficient system resources (Job Storage Block full) |
| IOX8 | 601212 | Monitor internal error |
| IOX9 | 601216 | Function legal for sequential write only |
| IPARP1 | 603460 | Cannot start ARP until TCPNI service is running |
| IPCF10 | 601027 | WHEEL capability required |
| IPCF11 | 601030 | WHEEL or IPCF capability required |
| IPCF12 | 601031 | No free PID's available |
| IPCF13 | 601032 | PID quota exceeded |
| IPCF14 | 601033 | No PID's available to this job |
| IPCF15 | 601034 | No PID's available to this process |
| IPCF16 | 601035 | Receive and message data modes do not match |
| IPCF17 | 601036 | Argument block too small |
| IPCF18 | 601037 | Invalid MUTIL JSYS function |
| IPCF19 | 601040 | No PID for [SYSTEM] INFO |
| IPCF20 | 601041 | Invalid process handle |
| IPCF21 | 601042 | Invalid job number |
| IPCF22 | 601043 | Invalid software interrupt channel number |
| IPCF23 | 601044 | [SYSTEM] INFO already exists |
| IPCF24 | 601045 | Invalid message size |
| IPCF25 | 601046 | PID does not belong to this job |
| IPCF26 | 601047 | PID does not belong to this process |
| IPCF27 | 601050 | PID is not defined |
| IPCF28 | 601051 | PID not accessible by this process |
| IPCF29 | 601052 | PID already being used by another process |
| IPCF30 | 601053 | Job is not logged in |
| IPCF31 | 601102 | Invalid page number |
| IPCF32 | 601103 | Page is not private |
| IPCF33 | 601130 | Invalid index into system PID table |
| IPCF34 | 601320 | Cannot receive into an existing page |
| IPCF35 | 602125 | Invalid IPCF quota |
| IPCF36 | 605000 | PID not assigned on this LCS processor |
| IPCFX1 | 601016 | Length of packet descriptor block cannot be less than 4 |
| IPCFX2 | 601017 | No message for this PID |
| IPCFX3 | 601020 | Data too long for user's buffer |
| IPCFX4 | 601021 | Receiver's PID invalid |
| IPCFX5 | 601022 | Receiver's PID disabled |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| IPCFX6 | 601023 | Send quota exceeded |
| IPCFX7 | 601024 | Receiver quota exceeded |
| IPCFX8 | 601025 | IPCF free space exhausted |
| IPCFX9 | 601026 | Sender's PID invalid |
| IPFLAD | 603456 | Local Internet host number not in GHT |
| IPHCHK | 603451 | Computed GHT checksum does not match |
| IPHCNT | 603452 | GHT entry count argument is not correct |
| IPHEMX | 603454 | Exceeded maximum number of GHT entries |
| IPHNSP | 603453 | Insufficient system resources (No free space for GHT) |
| IPHSEQ | 603455 | GHT Internet host numbers not in ascending order |
| KDPX01 | 602206 | KMC11 not running |
| KFRKX1 | 600365 | Illegal to kill top level process |
| KFRKX2 | 600366 | Illegal to kill self |
| KLPX1 | 602547 | No BHDs available |
| KLPX10 | 602734 | Don't know our CI node number |
| KLPX11 | 602735 | Queue is empty |
| KLPX12 | 602746 | Virtual circuit is not closed |
| KLPX13 | 602752 | Named Buffer transfer error |
| KLPX14 | 602754 | Timed out waiting for KLIPA disable to complete |
| KLPX15 | 602755 | Timed out waiting for KLIPA enable to complete |
| KLPX2 | 602550 | No BSDs available |
| KLPX3 | 602551 | No datagrams buffers available |
| KLPX4 | 602552 | No message buffers available |
| KLPX5 | 602727 | KLIPA is not enabled |
| KLPX6 | 602730 | KLIPA is in maintenance mode |
| KLPX7 | 602731 | No KLIPA on system |
| KLPX8 | 602732 | Packet is bad |
| KLPX9 | 602733 | No virtual circuit |
| LATX01 | 605010 | Buffer size too small for available data |
| LATX02 | 605011 | LAT parameter value out of range |
| LATX03 | 605012 | LAT is not operational |
| LATX04 | 605013 | Invalid or unknown LAT server name |
| LATX05 | 605014 | Invalid LAT parameter |
| LATX06 | 605015 | Invalid LAT parameter value |
| LATX07 | 605016 | Invalid or unknown LAT service name |
| LATX08 | 605017 | Insufficient LAT Resources |
| LATX09 | 605020 | LAT Host name already set |
| LATX10 | 605021 | Invalid or unknown LAT port name |
| LATX11 | 605022 | Invalid or unknown LAT connect id |
| LCBDBP | 601475 | Bad byte pointer passed to LCS |
| LCLNER | 601476 | LCS length error |
| LCNOND | 601477 | LCS No such node |
| LGINX1 | 600010 | Invalid account identifier |
| LGINX2 | 600011 | Directory is "files-only" and cannot be logged in to |
| LGINX3 | 600012 | Internal format of directory is incorrect |
| LGINX4 | 600013 | Invalid password |
| LGINX5 | 600014 | Job is already logged in |
| LGINX6 | 601337 | No more job slots available for logging-in |
| LLMX01 | 604000 | Transmit Datagram Failed |
| LLMX02 | 604001 | LLMOP State is OFF |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|---------|--------|---|
| LLMX03 | 604002 | Invalid byte pointer |
| LLMX04 | 604003 | Nonexistent Request Number |
| LLMX05 | 604004 | Invalid KLNI channel specified |
| LLMX06 | 604005 | Configurator interrupts assigned to another process |
| LLMX99 | 604777 | LLMOP Internal Error |
| LNGFX1 | 601317 | Page table does not exist and file not open for write |
| LNSTX1 | 601002 | No such logical name |
| LNSTX2 | 601136 | Invalid function |
| LOCKX1 | 601771 | Illegal to lock other than a private page |
| LOCKX2 | 601772 | Requested page unavailable |
| LOCKX3 | 601773 | Attempt to lock too much memory |
| LOUTX1 | 600035 | Illegal to specify job number when logging out own job |
| LOUTX2 | 600036 | Invalid job number |
| LOUTX3 | 601227 | WHEEL or OPERATOR capability required |
| LOUTX4 | 601230 | LOG capability required |
| LOUTX5 | 601753 | Illegal to log out job 0 |
| LPINX1 | 601333 | Invalid unit number |
| LPINX2 | 601334 | WHEEL or OPERATOR capability required |
| LPINX3 | 601335 | Illegal to load RAM or VFU while device is OPEN |
| LSTRX1 | 601405 | Process has not encountered any errors |
| LTLBLX | 602347 | Too many user labels |
| LTLBX1 | 602350 | Undefined record format on non-TOPS20 tape |
| METRX1 | 602352 | METER% not supported on this processor |
| MLK BX1 | 601003 | Lock facility already in use |
| MLK BX2 | 601004 | Too many pages to be locked |
| MLK BX3 | 601005 | Page is not available |
| MLK BX4 | 601006 | Illegal to remove previous contents of user map |
| MNTX1 | 600345 | Internal format of directory is incorrect |
| MNTX2 | 600346 | Device is not on line |
| MNTX3 | 600347 | Device is not mountable |
| MONX01 | 601727 | Insufficient system resources |
| MONX02 | 601730 | Insufficient system resources (JSB full) |
| MONX03 | 601731 | Monitor internal error |
| MONX04 | 601732 | Insufficient system resources (Swapping space full) |
| MONX05 | 602032 | Insufficient system resources (no resident free space) |
| MONX06 | 602433 | Insufficient system resources (No swappable free space) |
| MONX07 | 602553 | Insufficient system resources (no DECnet free space) |
| MREQ10 | 602261 | Density mismatch between request and volume |
| MREQ11 | 602262 | Drive type mismatch between request and volume |
| MREQ12 | 602263 | Label type mismatch between request and volume |
| MREQ13 | 602264 | Structural error in mount message |
| MREQ14 | 602265 | Setname mismatch between request and volume |
| MREQ15 | 602266 | Mount refused by operator |
| MREQ16 | 602267 | Volume identifiers not supplied by operator |
| MREQ17 | 602270 | Volume-identifier list missing |
| MREQ18 | 602271 | End of volume-identifier list reached while reading |
| MREQ19 | 602272 | Requested tape drive type not available to system |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| MREQ20 | 602273 | Structural error in mount entry |
| MREQ21 | 602274 | Mount requested for unknown device type |
| MREQ22 | 602311 | Structure name not specified |
| MREQ23 | 602344 | Dismount refused by operator |
| MREQ24 | 602345 | Illegal to dismount connected structure |
| MREQ25 | 602346 | Structure not found |
| MREQ26 | 602351 | Tape mounting function disabled by installation |
| MREQ27 | 602402 | Structure is set IGNORED |
| MREQ28 | 602403 | Cannot overwrite volume - first file is not expired |
| MREQ29 | 602404 | Cannot overwrite volume - write access required |
| MREQ30 | 602405 | Tape label format error |
| MREQ31 | 602430 | Insufficient MOUNTR resources |
| MREQX1 | 602250 | Request canceled by user |
| MREQX2 | 602251 | Labeled tapes not permitted on 7-track drives |
| MREQX3 | 602252 | Unknown density specified |
| MREQX4 | 602253 | Unknown drive type specified |
| MREQX5 | 602254 | Unknown label type specified |
| MREQX6 | 602255 | Set name illegal or not specified |
| MREQX7 | 602256 | Illegal starting-volume specification |
| MREQX8 | 602257 | Attempt to switch to volume outside set |
| MREQX9 | 602260 | Illegal volume identifier specified |
| MSCPX1 | 600517 | No MSCP server in current monitor |
| MSCPX2 | 600520 | Drive type error |
| MSCPX3 | 600521 | Requested drive not found |
| MSCPX4 | 605502 | MSCP server not currently running |
| MSTRX1 | 601345 | Invalid function |
| MSTRX2 | 601346 | WHEEL or OPERATOR capability required |
| MSTRX3 | 601347 | Argument block too small |
| MSTRX4 | 601350 | Insufficient system resources |
| MSTRX5 | 601351 | Drive is not on-line |
| MSTRX6 | 601352 | Home blocks are bad |
| MSTRX7 | 601353 | Invalid structure name |
| MSTRX8 | 601354 | Could not get OFN for ROOT-DIRECTORY |
| MSTRX9 | 601355 | Could not MAP ROOT-DIRECTORY |
| MSTX10 | 601356 | ROOT-DIRECTORY bad |
| MSTX11 | 601357 | Could not initialize Index Table |
| MSTX12 | 601360 | Could not OPEN Bit Table File |
| MSTX13 | 601361 | Backup copy of ROOT-DIRECTORY is bad |
| MSTX14 | 601362 | Invalid channel number |
| MSTX15 | 601363 | Invalid unit number |
| MSTX16 | 601364 | Invalid controller number |
| MSTX17 | 601421 | All units in a structure must be of the same type |
| MSTX18 | 601422 | No more units in system |
| MSTX19 | 601423 | Unit is already part of a mounted structure |
| MSTX20 | 601424 | Data error reading HOME blocks |
| MSTX21 | 601425 | Structure is not mounted |
| MSTX22 | 601426 | Illegal to change specified bits |
| MSTX23 | 601430 | Could not write HOME blocks |
| MSTX24 | 601750 | Illegal to dismount the System Structure |
| MSTX25 | 601751 | Invalid number of swapping pages |
| MSTX26 | 601752 | Invalid number of Front-End-Filesystem pages |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| MSTX27 | 601757 | Specified unit is not a disk |
| MSTX28 | 601760 | Could not initialize bit table for structure |
| MSTX29 | 601761 | Could not reconstruct ROOT-DIRECTORY |
| MSTX30 | 601770 | Incorrect Bit Table counts on structure |
| MSTX31 | 602000 | Structure already mounted |
| MSTX32 | 602001 | Structure was not mounted |
| MSTX33 | 602002 | Structure is unavailable for mounting |
| MSTX34 | 602063 | Unit is write-locked |
| MSTX35 | 602201 | Too many units in structure |
| MSTX36 | 602223 | Illegal while JFNs assigned |
| MSTX37 | 602224 | Illegal while connected to structure |
| MSTX40 | 602225 | Invalid PSI channel number given |
| MSTX41 | 601461 | Channel does not exist |
| MSTX42 | 601462 | Controller does not exist |
| MSTX43 | 603033 | Illegal to dismount structure during initialization |
| MSTX44 | 605001 | Mount type refused by another CFS processor |
| MSTX45 | 602615 | Structure naming or drive serial number conflict in CFS cluster |
| MSTX46 | 602616 | Illegal to specify mount attribute |
| MSTX47 | 601525 | Shared access denied; already set exclusive in CFS cluster |
| MSTX48 | 601526 | Exclusive access denied; access conflict in CFS cluster |
| MSTX49 | 601527 | Structure naming conflict in CFS cluster |
| MSTX50 | 601531 | Mount type refused by this CFS processor |
| MSTX51 | 601532 | Insufficient system resources (structure limit exceeded) |
| MTNX01 | 601514 | Serial number out of range |
| MTOX1 | 601210 | Invalid function |
| MTOX10 | 601323 | VFU or RAM file cannot be OPENed |
| MTOX11 | 601324 | Data too large for buffers |
| MTOX12 | 601325 | Input error or not all data read |
| MTOX13 | 601326 | Argument block too small |
| MTOX14 | 601327 | Invalid software interrupt channel number |
| MTOX15 | 601331 | Device does not have Direct Access (programmable) VFU |
| MTOX16 | 601332 | VFU or Translation Ram file must be on disk |
| MTOX17 | 601336 | Device is not on line |
| MTOX18 | 601407 | Invalid software interrupt channel number |
| MTOX19 | 601755 | Invalid terminal page width |
| MTOX2 | 601220 | Record size was not set before I/O was done |
| MTOX20 | 601756 | Invalid terminal page length |
| MTOX21 | 602753 | Illegal two character escape sequence |
| MTOX3 | 601221 | Function not legal in dump mode |
| MTOX4 | 601222 | Invalid record size |
| MTOX5 | 601213 | Invalid hardware data mode for magnetic tape |
| MTOX6 | 601223 | Invalid magnetic tape density |
| MTOX7 | 601226 | WHEEL or OPERATOR capability required |
| MTOX8 | 601274 | Argument block too long |
| MTOX9 | 601322 | Output still pending |
| NIEANE | 605424 | Address Not Enabled |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| NIECAB | 605436 | Command abort |
| NIECCF | 605431 | Carrier check failed |
| NIECIO | 605501 | Channel is owned by another fork |
| NIEDNS | 605422 | Datagram Not Sent |
| NIEEXC | 605421 | Excessive Collisions |
| NIEIBP | 605420 | Illegal Byte Pointer |
| NIEIBS | 605413 | Illegal Buffer Size |
| NIEICA | 605426 | Illegal Channel Address |
| NIEICS | 605435 | Illegal channel state |
| NIEIFB | 605412 | Improperly Formatted Buffer |
| NIEIMA | 605425 | Illegal Multicast Address |
| NIEIVP | 605405 | Illegal Protocol Type |
| NIELER | 605416 | Length Error |
| NIENPE | 605417 | No Protocol Type Enabled For This Portal |
| NIENRE | 605423 | No Room For Entry |
| NIENSC | 605403 | No Such Channel |
| NIENSP | 605411 | No Such Portal |
| NIEOPN | 605433 | Open circuit |
| NIEPIU | 605406 | Protocol Type In Use |
| NIEPRA | 605407 | Promiscuous Receiver Active |
| NIEPWS | 605427 | Portal in Wrong State |
| NIERAB | 605415 | Receive Aborted |
| NIERDL | 605414 | Received Datagram Too Long |
| NIERFD | 605434 | Remote failure to defer |
| NIERTE | 605500 | Receive or Transmit quota exceeded |
| NIESHT | 605432 | Short circuit |
| NODX01 | 602115 | Node name exceeds 6 characters |
| NODX02 | 602207 | Line not turned off |
| NODX03 | 602210 | Another line already looped |
| NODX04 | 602576 | No local node name defined |
| NODX05 | 602577 | Function no longer supported |
| NODX06 | 602600 | Resource allocation failure |
| NODX07 | 602647 | Argument block not long enough |
| NODX10 | 602650 | Channel number out of range |
| NODX11 | 602651 | Job number out of range |
| NODX12 | 602700 | Bad table designator |
| NODX13 | 602701 | Bad 1st argument |
| NODX14 | 602702 | Bad 2nd argument |
| NODX15 | 602703 | No such table |
| NODX16 | 602756 | DECnet has already initialized |
| NODX17 | 602766 | Illegal parameter value |
| NOUTX1 | 600407 | Radix is not in range 2 to 36 |
| NOUTX2 | 600410 | Column overflow |
| NPX2CL | 602413 | Two colons required on node name |
| NPXAMB | 602044 | Ambiguous |
| NPXCMA | 602057 | Comma not given |
| NPXICN | 602052 | Invalid character in number |
| NPXIDT | 602053 | Invalid device terminator |
| NPXINW | 602050 | Invalid guide word |
| NPXNC | 602051 | Not confirmed |
| NPXNMD | 602056 | Does not match directory or user name, or structure |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| | | not mounted |
| NPXNMT | 602055 | Does not match token |
| NPXNOM | 602046 | Does not match switch or keyword |
| NPXNQS | 602054 | Not a quoted string - quote missing at beginning or end |
| NPXNSW | 602045 | Not a switch - does not begin with slash |
| NPXNUL | 602047 | Null switch or keyword given |
| NSJX01 | 602555 | WHEEL or OPERATOR capability required |
| NSJX02 | 602556 | Allocation failure |
| NSJX03 | 602557 | Wrong number of arguments |
| NSJX04 | 602560 | Illegal function |
| NSJX05 | 602561 | Connect block length error |
| NSJX06 | 602562 | Address Error |
| NSJX07 | 602563 | Argument Block Format Error |
| NSJX08 | 602564 | Process block length error |
| NSJX09 | 602565 | Bad format type in process block |
| NSPX00 | 602353 | Reject or disconnect by object |
| NSPX01 | 602354 | Resource allocation failure |
| NSPX02 | 602355 | Destination node does not exist |
| NSPX03 | 602356 | Remote node shutting down |
| NSPX04 | 602357 | Destination process does not exist |
| NSPX05 | 602360 | Invalid process name field |
| NSPX06 | 602361 | Object is busy |
| NSPX07 | 602362 | Unspecified error |
| NSPX08 | 602363 | Abort by management |
| NSPX09 | 602364 | Abort by object |
| NSPX10 | 602365 | Flow control violation |
| NSPX11 | 602366 | Too many connections to node |
| NSPX12 | 602367 | Too many connections to destination process |
| NSPX13 | 602370 | Access not permitted |
| NSPX14 | 602371 | Logical link services mismatch |
| NSPX15 | 602372 | Invalid account |
| NSPX16 | 602373 | SEGSIZE too small |
| NSPX17 | 602374 | No response from destination process |
| NSPX18 | 602375 | Node unreachable |
| NSPX19 | 602376 | Link aborted due to data loss |
| NSPX20 | 602377 | Destination process does not exist |
| NSPX21 | 602400 | Confirmation of DI |
| NSPX22 | 602401 | Image data field too long |
| NSPX23 | 602411 | Invalid NSP reason code |
| NSPX24 | 602456 | Node name not assigned to a network node |
| NSPX25 | 602457 | Illegal DECnet node number |
| NSPX26 | 602460 | Table of topology watchers is full |
| NSPX27 | 602545 | Local node shut |
| NTMX1 | 602451 | Network Management unable to complete request |
| NTMX2 | 602751 | Event resource already in use |
| NTMX3 | 602765 | DECnet is not initialized |
| NTWZX1 | 600737 | NET WIZARD capability required |
| ODTNX1 | 600462 | Time zone must be USA or Greenwich |
| OPNX1 | 600120 | File is already open |
| OPNX10 | 600131 | Entire file structure full |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| OPNX12 | 600133 | List access required |
| OPNX13 | 600134 | Invalid access requested |
| OPNX14 | 600135 | Invalid mode requested |
| OPNX15 | 600136 | Read/write access required |
| OPNX16 | 600137 | File has bad index block |
| OPNX17 | 600140 | No room in job for long file page table |
| OPNX18 | 600141 | Unit Record Devices are not available |
| OPNX19 | 600142 | IMP is not up |
| OPNX2 | 600121 | File does not exist |
| OPNX20 | 600143 | Host is not up |
| OPNX21 | 600144 | Connection refused |
| OPNX22 | 600145 | Connection byte size does not match |
| OPNX23 | 601132 | Disk quota exceeded |
| OPNX25 | 601224 | Device is write locked |
| OPNX26 | 601410 | Illegal to open a string pointer |
| OPNX3 | 600122 | Read access required |
| OPNX30 | 602326 | File has archive status, modification is prohibited |
| OPNX31 | 602327 | File is off line |
| OPNX4 | 600123 | Write access required |
| OPNX5 | 600124 | Execute access required |
| OPNX6 | 600125 | Append access required |
| OPNX7 | 600126 | Device already assigned to another job |
| OPNX8 | 600127 | Device is not on line |
| OPNX9 | 600130 | Invalid simultaneous access |
| PAGPTN | 601530 | Page table entry nonzero. (DEC internal error code.) |
| PDVX01 | 601554 | Address in .POADE must be as large as address in .POADR |
| PDVX02 | 601555 | Addresses in .PODAT block must be in strict ascending order |
| PDVX03 | 601556 | Address in .POADR must be a program data vector address |
| PEEKX2 | 600617 | Read access failure on monitor page |
| PMAPX1 | 600240 | Invalid access requested |
| PMAPX2 | 600241 | Invalid use of PMAP |
| PMAPX3 | 601104 | Illegal to move shared page into file |
| PMAPX4 | 601105 | Illegal to move file page into process |
| PMAPX5 | 601106 | Illegal to move special page into file |
| PMAPX6 | 601107 | Disk quota exceeded |
| PMAPX7 | 601415 | Illegal to map file on dismounted structure |
| PMAPX8 | 602464 | Indirect page map loop detected |
| PMCLX1 | 602005 | Illegal page state or state transition |
| PMCLX2 | 602006 | Requested physical page is unavailable |
| PMCLX3 | 602007 | Requested physical page contains errors |
| PMCLX4 | 602165 | No more error information |
| PPNX1 | 601444 | Invalid PPN |
| PPNX2 | 601445 | Structure is not mounted |
| PPNX3 | 601446 | Insufficient system resources |
| PPNX4 | 601447 | Invalid directory number |
| PRAX1 | 601263 | Invalid PRARG function code |
| PRAX2 | 601264 | No room in monitor data base for argument block |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| PRAX3 | 601270 | PRARG argument block too large |
| QUEUX1 | 601504 | Illegal argument list passed to QUEUE% |
| QUEUX2 | 601505 | Invalid function |
| QUEUX3 | 601506 | Fatal error returned from application |
| QUEUX4 | 601507 | Invalid message returned from ORION |
| QUEUX5 | 601510 | Insufficient system resources (Job Storage Block full) |
| QUEUX6 | 601511 | Illegal response length |
| QUEUX7 | 601512 | Argument block too small |
| RCDIX1 | 601376 | Insufficient system resources |
| RCDIX2 | 601377 | Invalid directory specification |
| RCDIX3 | 601400 | Invalid structure name |
| RCDIX4 | 601401 | Monitor internal error |
| RCUSX1 | 601402 | Insufficient system resources |
| RDDIX1 | 600560 | Illegal to read directory for this device |
| RDTX1 | 601010 | Invalid string pointer |
| RFRKX1 | 600367 | Processes are not frozen |
| RIRX1 | 602426 | RIR JSYS incompatible with previous XSIR |
| RJFNX1 | 600165 | File is not closed |
| RJFNX2 | 600166 | JFN is being used to accumulate filename |
| RJFNX3 | 600167 | JFN is not accessible by this process |
| RNAMX1 | 600450 | Files are not on same device |
| RNAMX2 | 600451 | Destination file expunged |
| RNAMX3 | 600452 | Write or owner access to destination file required |
| RNAMX4 | 600453 | Quota exceeded in destination of rename |
| RNAMX5 | 600750 | Destination file is not closed |
| RNAMX6 | 600751 | Destination file has bad page table |
| RNAMX7 | 600752 | Source file expunged |
| RNAMX8 | 600753 | Write or owner access to source file required |
| RNAMX9 | 600754 | Source file is nonexistent |
| RNMX10 | 600755 | Source file is not closed |
| RNMX11 | 600756 | Source file has bad page table |
| RNMX12 | 600757 | Illegal to rename to self |
| RNMX13 | 601454 | Insufficient system resources |
| RSCNX1 | 600361 | Overflowed rescan buffer, input string truncated |
| RSCNX2 | 600362 | Invalid function code |
| RUNTX1 | 600273 | Invalid process handle -3 or -4 |
| SACTX1 | 600530 | File is not on multiple-directory device |
| SACTX2 | 600531 | Insufficient system resources (Job Storage Block full) |
| SACTX3 | 600532 | Directory requires numeric account |
| SACTX4 | 600533 | Write or owner access required |
| SAVX1 | 601330 | Illegal to save files on this device |
| SCAPTL | 602602 | Message too long |
| SCLX01 | 602652 | No connect data to read |
| SCLX02 | 602653 | Percentage input out of bounds |
| SCLX03 | 602654 | Function called in wrong state |
| SCLX04 | 602655 | Unexpected state - disconnect sent |
| SCLX05 | 602656 | Unexpected state - disconnect confirmed |
| SCLX06 | 602657 | Unexpected state - no confidence |
| SCLX07 | 602660 | Unexpected state - no link |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| SCLX08 | 602661 | Unexpected state - no communication |
| SCLX09 | 602662 | Unexpected state - no resources |
| SCLX10 | 602663 | Unrecognized object |
| SCLX11 | 602664 | Object too busy |
| SCLX12 | 602665 | Disconnect complete |
| SCLX13 | 602666 | Image field too long |
| SCLX14 | 602667 | Unspecified reject reason |
| SCLX15 | 602670 | Bad combination of SAEOM & SAWAI flags |
| SCLX16 | 602671 | Address error in user argument |
| SCLX17 | 602672 | Illegal message format detected |
| SCLX18 | 602673 | Unexpected state - connect wait |
| SCLX19 | 602674 | Unexpected state - connect received |
| SCLX20 | 602675 | Unexpected state - connect sent |
| SCLX21 | 602676 | Unexpected state - reject |
| SCLX22 | 602677 | Unexpected state - run |
| SCSAAB | 602571 | Error accessing argument block |
| SCSBAS | 602716 | Internal error, bad argument to subroutine |
| SCSBFC | 602566 | Function code out of range |
| SCSBTS | 602567 | Argument block too short |
| SCSCWS | 602714 | Connection in incorrect state for function |
| SCSDCB | 602575 | Datagram send text crosses a page boundry |
| SCSDTL | 602611 | DMA buffer to long |
| SCSENB | 602704 | Excessive number of buffers in queue request |
| SCSFRK | 602617 | Fork does not own this SCS% data |
| SCSIAA | 602623 | Invalid address in arguments |
| SCSIAB | 602570 | Invalid argument block address |
| SCSIBN | 602764 | Invalid buffer name |
| SCSIBP | 602624 | Invalid byte pointer |
| SCSIDM | 602726 | Invalid DMA transmission mode |
| SCSIFL | 602724 | Invalid forward link in buffer chain |
| SCSIID | 602603 | Invalid connect ID |
| SCSIPC | 602722 | PSI channel out of range |
| SCSIPS | 602723 | Invalid path spec |
| SCSISB | 602621 | Invalid node number |
| SCSIST | 602725 | Invalid SCS% interrupt type |
| SCSJBD | 602646 | No user address found for sent packet |
| SCSNBA | 602605 | Internal resources exhausted (No more SCA buffers) |
| SCSNDQ | 602645 | No datagram buffers queued |
| SCSNEB | 602720 | Insufficient buffers to fill request |
| SCSNEC | 602715 | Not enough credit |
| SCSNEP | 602573 | Not enough privileges enabled |
| SCSNKP | 602721 | No known KLIPA on this system |
| SCSNMQ | 602620 | No buffers queued for message reception |
| SCSNPA | 602604 | No packet address |
| SCSNRT | 602601 | No room in table for address entry |
| SCSNSB | 602717 | No such buffer |
| SCSNSC | 602574 | No such connect ID |
| SCSNSD | 602610 | No such DMA buffer name |
| SCSNSH | 602622 | Not enough room for SCA headers |
| SCSNSN | 602572 | No source process name specified on connection request |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| SCSQIE | 602613 | Queue is empty |
| SCSSCP | 602607 | DMA segment crosses a page boundary |
| SCSSTL | 602710 | DMA buffer segment too long |
| SCSTBF | 601524 | No slots left in CID tables |
| SCSTMS | 602711 | Too many DMA buffer segments |
| SCSUPC | 602612 | Unknown PSI code |
| SCSZLP | 602606 | Zero length packet text |
| SCTX1 | 601550 | Invalid function code |
| SCTX2 | 601551 | Terminal already in use as controlling terminal |
| SCTX3 | 601552 | Illegal to redefine the job's controlling terminal |
| SCTX4 | 601553 | SC%SCT capability required |
| SEVEX1 | 600610 | Entry vector length is not less than 1000 |
| SFBSX1 | 600210 | Illegal to change byte size for this opening of file |
| SFBSX2 | 600211 | Invalid byte size |
| SFPTX1 | 600175 | File is not open |
| SFPTX2 | 600176 | Illegal to reset pointer for this file |
| SFPTX3 | 600177 | Invalid byte number |
| SFRVX1 | 600377 | Invalid position in entry vector |
| SFUSX1 | 601373 | Invalid function |
| SFUSX2 | 601374 | Insufficient system resources |
| SFUSX3 | 601375 | No such user name |
| SFUSX4 | 601700 | File expunged |
| SFUSX5 | 601701 | Write or owner access required |
| SFUSX6 | 601702 | No such user name |
| SIRX1 | 600570 | Table address is not greater than 20 |
| SIRX2 | 602425 | SIR JSYS invoked from non-zero section |
| SJBX1 | 601244 | Invalid function |
| SJBX2 | 601245 | Invalid magnetic tape density |
| SJBX3 | 601246 | Invalid magnetic tape data mode |
| SJBX4 | 601251 | Invalid job number |
| SJBX5 | 601252 | Job is not logged in |
| SJBX6 | 601253 | WHEEL or OPERATOR capability required |
| SJBX7 | 602077 | Remark exceeds 39 characters |
| SJBX8 | 601455 | Illegal to perform this function |
| SJPRX1 | 601276 | Job is not logged in |
| SKDX1 | 602247 | Cannot change class |
| SMAPX1 | 602431 | Attempt to delete a section still shared |
| SMAPX2 | 602465 | Indirect section map loop detected |
| SMONX1 | 600516 | WHEEL or OPERATOR capability required |
| SMONX2 | 601250 | Invalid SMON function |
| SMONX3 | 601677 | Timeout interval out of range |
| SMONX4 | 605630 | Minimum password length must be between 1 and 39 characters |
| SMONX5 | 605634 | ACJ fork already running |
| SMONX6 | 605635 | Invalid request |
| SMONX7 | | Password expiration day count must be between 1 and 366 |
| SNDIX1 | 600732 | Invalid message size |
| SNDIX2 | 600733 | Insufficient system resources (No buffers available) |
| SNDIX3 | 600734 | Illegal to specify NCP links 0 - 72 |
| SNDIX4 | 600735 | Invalid header value for this queue |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| SNDIX5 | 600736 | IMP down |
| SNOP10 | 601121 | Breakpoints already inserted |
| SNOP11 | 601122 | Breakpoints not inserted |
| SNOP12 | 601123 | Invalid format for program name symbol |
| SNOP13 | 601124 | No such program name symbol |
| SNOP14 | 601125 | No such symbol |
| SNOP15 | 601126 | Not enough free pages for snooping |
| SNOP16 | 601127 | Multiply defined symbol |
| SNOP17 | 601131 | Breakpoint already defined |
| SNOP18 | 601163 | Data page is not private or copy-on-write |
| SNOPX1 | 601110 | WHEEL or OPERATOR capability required |
| SNOPX2 | 601111 | Invalid function |
| SNOPX3 | 601112 | .SNPLC function must be first |
| SNOPX4 | 601113 | Only one .SNPLC function allowed |
| SNOPX5 | 601114 | Invalid page number |
| SNOPX6 | 601115 | Invalid number of pages to lock |
| SNOPX7 | 601116 | Illegal to define breakpoints after inserting them |
| SNOPX8 | 601117 | Breakpoint is not set on instruction |
| SNOPX9 | 601120 | No more breakpoints allowed |
| SPACX1 | 600245 | Invalid access requested |
| SPLBFC | 600264 | Bad function code |
| SPLBTS | 600263 | Argument block too short |
| SPLFX1 | 600260 | Process is not inferior or equal to self |
| SPLFX2 | 600261 | Process is not inferior to self |
| SPLFX3 | 600262 | New superior process is inferior to intended inferior |
| SPLX1 | 601144 | Invalid function |
| SPLX2 | 601145 | Argument block too small |
| SPLX3 | 601146 | Invalid device designator |
| SPLX4 | 601147 | WHEEL or OPERATOR capability required |
| SPLX5 | 601150 | Illegal to specify 0 as generation number for first file |
| SPLX6 | 601450 | No directory to write spooled files into |
| SQX1 | 600742 | Special network queue handle out of range |
| SQX2 | 600743 | Special network queue not assigned |
| SSAVX1 | 600600 | Illegal to save files on this device |
| SSAVX2 | 600601 | Page count (left half of table entry) must be negative |
| SSAVX3 | 601232 | Insufficient system resources (Job Storage Block full) |
| SSAVX4 | 601233 | Directory area of EXE file is more than one page |
| SSAVX5 | 601500 | Number of PDVs grew during save |
| STADX1 | 600275 | WHEEL or OPERATOR capability required |
| STADX2 | 600276 | Invalid date or time |
| STDIX1 | 602003 | The STDIR JSYS has been replaced by RCDIR and RCUSR |
| STDVX1 | 600332 | No such device |
| STRX01 | 601436 | Structure is not mounted |
| STRX02 | 601437 | Insufficient system resources |
| STRX03 | 601442 | No such directory name |
| STRX04 | 601443 | Ambiguous directory specification |
| STRX05 | 601434 | No such user name |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|---|
| STRX06 | 601747 | No such user number |
| STRX07 | 602142 | Invalid user number |
| STRX08 | 602143 | Invalid user name |
| STRX09 | 602222 | Prior structure mount required |
| STRX10 | 601676 | Structure is offline |
| STRX11 | 601674 | Invalid structure number |
| STYPX1 | 601414 | Invalid terminal type |
| SWJFX1 | 601406 | Illegal to swap same JFN |
| SWJFX2 | 602242 | Illegal to swap ATS JFN |
| SYEX1 | 601206 | Unreasonable SYSERR block size |
| SYEX2 | 601207 | No buffer space available for SYSERR |
| TADDX1 | 601235 | Table is full |
| TADDX2 | 601236 | Entry is already in table |
| TCPX10 | 603411 | Unable to decode TIMEOUT attribute in TCP specification |
| TCPX11 | 603412 | Unable to decode TYPE-OF-SERVICE attribute in TCP specification |
| TCPX12 | 603413 | Unable to decode SECURITY attribute in TCP specification |
| TCPX13 | 603414 | Unable to decode COMPARTMENTS attribute in TCP specification |
| TCPX14 | 603415 | Unable to decode HANDLING-RESTRICTIONS attribute in TCP specification |
| TCPX15 | 603416 | Unable to decode TRANSMISSION-CONTROL attribute in TCP specification |
| TCPX16 | 603417 | TCP not initialized and available |
| TCPX17 | 603420 | Illegal IO mode for TCP device |
| TCPX18 | 603421 | Illegal byte size for TCP device |
| TCPX19 | 603422 | TCP connection allready exists |
| TCPX20 | 603423 | Maximum TCP connections exceeded |
| TCPX21 | 603424 | Wheel, Operator, or Network Wizard needed for special TCOPR function |
| TCPX22 | 603425 | Invalid TCOPR function requested |
| TCPX23 | 603426 | Invalid IPOPR function requested |
| TCPX24 | 603427 | Wheel, Operator, or Network Wizard needed for special IPOPR function |
| TCPX25 | 603430 | Open failure |
| TCPX26 | 603431 | Illegal Persist parameters |
| TCPX27 | 603432 | Illegal TCOPR Function on an OPEN TCP JFN |
| TCPX28 | 603433 | Invalid BBN TCP JSYS call |
| TCPX29 | 603434 | Assigned JFN too large for TCPJFN |
| TCPX30 | 603435 | Illegal TCP IO mode |
| TCPX31 | 603436 | Connection error or connection rejected |
| TCPX32 | 603437 | Retransmission timeout |
| TCPX33 | 603440 | Connection closed or closing |
| TCPX34 | 603441 | TCOPR Argument |
| TCPX35 | 603442 | Illegal to reopen a TCP JFN |
| TCPX36 | 603443 | Illegal TCOPR Function on an UNOPEN TCP JFN |
| TCPX37 | 603444 | No free space for buffer |
| TCPX40 | 603445 | TCOPR Function not yet implemented |
| TCPX41 | 603446 | TCOPR DEC interrupt channels not off |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| TCPX42 | 603447 | TCOPR Invalid TCB offset |
| TCPX43 | 603450 | TCOPR Invalid argument block |
| TCPX44 | 603461 | Monitor does not support TCP over Ethernet |
| TCPXX1 | 603400 | No IP free space for TCB |
| TCPXX2 | 603401 | Unable to decode local side TCP of specification |
| TCPXX3 | 603402 | Unable to decode foreign side TCP of specification |
| TCPXX4 | 603403 | Generation found in TCP specification |
| TCPXX5 | 603404 | TCP specification attribute not known to TCP |
| TCPXX6 | 603405 | Unable to decode CONNECTION attribute in TCP specification |
| TCPXX7 | 603406 | Unable to decode FOREIGN-HOST attribute in TCP specification |
| TCPXX8 | 603407 | Unable to decode LOCAL-HOST attribute in TCP specification |
| TCPXX9 | 603410 | Unable to decode PERSIST attribute in TCP specification |
| TDELX1 | 601234 | Table is empty |
| TDELX2 | 601403 | Invalid table entry location |
| TERMX1 | 600350 | Invalid terminal code |
| TFRKX1 | 600375 | Undefined function code |
| TFRKX2 | 600376 | Unassigned fork handle or not immediate inferior |
| TFRKX3 | 600411 | Fork(s) not frozen |
| TILFX1 | 600465 | Invalid time format |
| TIMEX1 | 600460 | Time cannot be greater than 24 hours |
| TIMEX2 | 601302 | Downtime cannot be more than 7 days in the future |
| TIMX1 | 601157 | Invalid function |
| TIMX10 | 601541 | No system date and time |
| TIMX2 | 601160 | Invalid process handle |
| TIMX3 | 601161 | Time limit already set |
| TIMX4 | 601162 | Illegal to clear time limit |
| TIMX5 | 601404 | Invalid software interrupt channel number |
| TIMX6 | 601535 | Time has already passed |
| TIMX7 | 601536 | No space available for a clock |
| TIMX8 | 601537 | User clock allocation exceeded |
| TIMX9 | 601540 | No such clock entry found |
| TLNKX1 | 600351 | Illegal to set remote to object before object to remote |
| TLNKX2 | 600356 | Link was not received within 15 seconds |
| TLNKX3 | 600357 | Links full |
| TLUKX1 | 601237 | Internal format of table is incorrect |
| TMONX1 | 601247 | Invalid TMON function |
| TTMSX1 | 602432 | Could not send message within timeout interval |
| TTMSX2 | 602743 | User is refusing messages and/or links |
| TTMSX3 | 605605 | Invalid CI node number |
| TTMSX4 | 605626 | Remote node not accepting remote sendalls |
| TTYX01 | 602030 | Line is not active |
| TTYX02 | 602455 | Illegal character specified |
| TTYX03 | 602543 | Line is temporarily active |
| TTYX04 | 605602 | Job is detached |
| TTYX1 | 600360 | Device is not a terminal |
| UFPGX1 | 601316 | File is not open for write |

TOPS-20 ERROR CODES AND MNEMONICS

| | | |
|--------|--------|--|
| USGX01 | 602113 | Invalid USAGE entry type code |
| USGX02 | 602116 | Item not found in argument list |
| USGX03 | 602124 | Default item not allowed |
| USGX04 | 601675 | Invalid terminal line number |
| UTSTX1 | 602013 | Invalid function code |
| UTSTX2 | 602014 | Area of code too large to test |
| UTSTX3 | 602015 | UTEST facility in use by another process |
| VACCX0 | 602111 | Invalid account |
| VACCX1 | 602112 | Account string exceeds 39 characters |
| VACCX2 | 602126 | Account has expired |
| VBCX1 | 601007 | Display data area not locked in core |
| WHELX1 | 600614 | WHEEL or OPERATOR capability required |
| WILDY1 | 601460 | Second JFN cannot be wild |
| XPEK01 | 602744 | Illegal system fork number specified |
| XPEK02 | 602745 | Unassigned system fork number specified |
| XPEK03 | 602747 | Word count not positive |
| XPEK04 | 602750 | Word count too large. Can not cross section boundaries |
| XSEVX1 | 602472 | Illegal entry vector type |
| XSEVX2 | 602473 | Invalid entry vector length |
| XSEVX3 | 602474 | Cannot get extended values with this monitor call |
| XSIRX1 | 602424 | Channel table crosses section boundary |
| XSIRX2 | 602427 | Level table crosses section boundary |
| ZONEX1 | 600461 | Time zone out of range |

INDEX

-A-

AC's, 1-1, 1-2
 ACCES Directory-related JSYSSs,
 3-1
 ACCES JSYS, 3-1
 ACCES Structure-related JSYSSs,
 3-1
 Access control, 2-74, 3-132
 Access modes, 2-9
 Access-control functions, 2-74
 Access-control program, 3-146,
 3-461
 Accounting functions, 2-1
 Accounting JSYSSs, 3-523
 Accumulators, 1-1, 1-2
 Acquiring physical memory, 3-349
 ADBRK Debugging JSYSSs, 3-3
 ADBRK JSYS, 3-3
 Adding a table entry, 3-493
 Address, 1-9
 Global, 1-3
 Address breaks, 3-3
 Address global, 1-3
 Address section-relative, 1-3
 AIC JSYS, 3-7
 AIC Software-interrupt JSYSSs, 3-7
 ALLOC Device-related JSYSSs, 3-8
 ALLOC Job-related JSYSSs, 3-8
 ALLOC JSYS, 3-8
 ANSI ASCII mode, 2-47
 Append access, 2-9, 3-340
 ARCF Archive-related JSYSSs, 3-9
 ARCF JSYS, 3-9
 Archive/virtual disk system, 2-92
 Arguments JSYS, 1-2
 ARPAnet-related JSYSSs
 TCOPR%, 3-497
 ASCII strings, 1-8
 ASND Device-related JSYSSs, 3-13
 ASND JSYS, 3-13
 ASNIQ% JSYS, 3-13
 ASNSQ JSYS, 3-14
 ASNSQ TCP/IP-related JSYSSs, 3-14
 Assigning a device, 3-13
 Assigning devices, 3-380
 Assigning disk addresses, 3-97
 Assigning TCP/IP queue, 3-14

Assigning terminal interrupt,
 3-17
 ATACH Job-related JSYSSs, 3-15
 ATACH JSYS, 3-15
 ATACH Terminal-related JSYSSs,
 3-15
 ATI JSYS, 3-17
 ATI Software-interrupt JSYSSs,
 3-17
 ATI Terminal-related JSYSSs, 3-17
 ATNVT JSYS, 3-17
 ATNVT TCP/IP-related JSYSSs, 3-17
 Attaching a job, 3-15

-B-

Backing up pointer, 3-19
 BIN Byte-I/O JSYSSs, 3-18
 BIN I/O JSYSSs, 3-18
 BIN TTY-I/O JSYSSs, 3-18
 18-bit address, 1-3
 23-bit address, 1-3
 30-bit address, 1-3
 BKJFN File-related JSYSSs, 3-19
 BKJFN JSYS, 3-19
 BKJFN Terminal-related JSYSSs,
 3-19
 BOOT JSYS, 3-19
 BOUT Byte-I/O JSYSSs, 3-25
 BOUT I/O JSYSSs, 3-25
 BOUT JSYS, 3-25
 BOUT TTY-I/O JSYSSs, 3-25
 Buffered I/O, 2-42
 BUGCHK facility
 dump on, 3-94, 3-186
 Byte input, 3-18, 3-344
 Byte output, 3-25, 3-345
 Byte pointer, 1-5, 1-7, 1-8, 1-9
 Byte pointer local, 1-5
 Byte pointer one-word global, 1-5

-C-

CACCT Accounting JSYSSs, 3-25
 CACCT Job-related JSYSSs, 3-25
 CACCT JSYS, 3-25
 CACCT Parameter-setting JSYSSs,
 3-25

Capabilities, 2-72
 Capabilities functions, 2-72
 Carriage control tape, 2-39
 CCOC word, 2-52
 CCOC words, 3-383, 3-433
 CDP:, 2-35, 2-38, 2-61
 CDR:, 2-35, 2-36, 2-37, 2-61
 CFBIF File-related JSYSSs, 3-26
 CFBIF I/O JSYSSs, 3-26
 CFBIF JSYS, 3-26
 CFBIF Terminal-related JSYSSs,
 3-26
 CFBIF TTY-I/O JSYSSs, 3-26
 CFOBF File-related JSYSSs, 3-27
 CFOBF I/O JSYSSs, 3-27
 CFOBF JSYS, 3-27
 CFOBF Terminal-related JSYSSs,
 3-27
 CFOBF TTY-I/O JSYSSs, 3-27
 CFORK JSYS, 3-27
 CFORK Process-related JSYSSs, 3-27
 Changing account, 3-25
 Character editing, 2-2
 CHFDB File-related JSYSSs, 3-29
 CHFDB JSYS, 3-29
 CHFDB Parameter-setting JSYSSs,
 3-29
 CHKAC Directory-related JSYSSs,
 3-30
 CHKAC File-related JSYSSs, 3-30
 CHKAC Info-returning JSYSSs, 3-30
 CHKAC JSYS, 3-30
 CIS JSYS, 3-31
 CIS Process-related JSYSSs, 3-31
 CIS Software-interrupt JSYSSs,
 3-31
 Clearing file input buffer, 3-26
 Clearing file output buffer, 3-27
 Clearing software interrupt
 system, 3-31
 CLOSF Device-related JSYSSs, 3-31
 CLOSF File-related JSYSSs, 3-31
 CLOSF JSYS, 3-31
 Closing a file, 3-31
 Closing process files, 3-33
 CLZFF Device-related JSYSSs, 3-33
 CLZFF File-related JSYSSs, 3-33
 CLZFF JSYS, 3-33
 CLZFF Process-related JSYSSs, 3-33
 CNFIG% JSYS, 3-34
 Command parsing, 3-37
 Communications-related JSYSSs
 ASNIQ%, 3-13
 SCS%, 3-406
 SNDIN%, 3-465
 TCOPR%, 3-497
 COMND JSYS, 3-37
 COMND Numeric-I/O JSYSSs, 3-37
 COMND TTY-I/O JSYSSs, 3-37
 Comparing strings, 3-482, 3-524
 Compatibility Package, 2-80
 Compatibility package, 3-424
 Compatibility package entry
 vector, 3-424
 Configuration information
 CNFIG% JSYS, 3-34
 Converting internal date/time,
 3-334
 Converting to internal date/time,
 3-181
 CRDIR Directory-related JSYSSs,
 3-61
 CRDIR JSYS, 3-61
 Creating a logical name, 3-75
 Creating a new job, 3-68
 Creating a section, 3-455
 Creating an inferior process,
 3-27
 Creating NVT connection, 3-17
 Creating sections, 3-400
 CRJOB Job-related JSYSSs, 3-68
 CRJOB JSYS, 3-68
 CRLNM JSYS, 3-75
 CRLNM Logical-name JSYSSs, 3-75
 Current section, 1-3

 -D-
 Data-conversion functions, 2-86
 Date-and-time functions, 2-89
 Date/time conversion, 2-89
 Date/time format, 2-89
 DCN:, 2-35, 2-62
 Deactivating interrupt channels,
 3-92
 Deassigning terminal interrupt,
 3-101
 DEBRK JSYS, 3-76
 DEBRK Software-interrupt JSYSSs,
 3-76
 Deferred terminal interrupt, 2-70
 DELDF Archive-related JSYSSs, 3-76
 DELDF File-related JSYSSs, 3-76

DELDF JSYS, 3-76
 Deleting a table entry, 3-494
 Deleting files, 3-78, 3-79
 DELF Archive-related JSYSs, 3-78
 DELF File-related JSYSs, 3-78
 DELF JSYS, 3-78
 DELNF Archive-related JSYSs, 3-79
 DELNF File-related JSYSs, 3-79
 DELNF JSYS, 3-79
 DEQ ENQ/DEQ JSYSs, 3-80
 DEQ JSYS, 3-80
 Designator destination, 1-6
 Designator device, 1-6
 Designator source, 1-6
 Designator terminal, 1-6
 Destination designator, 1-6
 Detaching a job, 3-101
 Device allocation, 3-8
 Device Characteristics Word, 3-105
 Device designator, 1-6, 1-10
 Device functions, 2-6, 2-34
 Device opening a, 2-6
 Device-control functions, 3-257
 Devices, 2-34
 DEVST Device-related JSYSs, 3-82
 DEVST Info-returning JSYSs, 3-82
 DEVST JSYS, 3-82
 DFIN I/O JSYSs, 3-82
 DFIN JSYS, 3-82
 DFIN Numeric-I/O JSYSs, 3-82
 DFIN TTY-I/O JSYSs, 3-82
 DFOUT I/O JSYSs, 3-83
 DFOUT JSYS, 3-83
 DFOUT Numeric-I/O JSYSs, 3-83
 DFOUT TTY-I/O JSYSs, 3-83
 DIAG Device-related JSYSs, 3-84
 DIBE File-related JSYSs, 3-91
 DIBE JSYS, 3-91
 DIBE Process-related JSYSs, 3-91
 DIBE Software-interrupt JSYSs, 3-91
 DIC JSYS, 3-92
 DIC Software-interrupt JSYSs, 3-92
 DIR Software-interrupt JSYSs, 3-92
 Directory access, 2-10, 3-1
 DIRST Directory-related JSYSs, 3-93
 DIRST Info-returning JSYSs, 3-93
 DIRST JSYS, 3-93
 Disabling interrupt system, 3-92
 Dismissing a process, 3-91, 3-94, 3-97, 3-523
 Dismissing an interrupt, 2-70
 Dismissing interrupt, 3-76
 DISMS JSYS, 3-94
 DISMS Process-related JSYSs, 3-94
 DOB% interface, 3-94
 DOB% JSYS, 3-94
 DOBE File-related JSYSs, 3-97
 DOBE JSYS, 3-97
 DOBE Process-related JSYSs, 3-97
 DOBE Software-interrupt JSYSs, 3-97
 Double-precision input, 3-82
 Double-precision output, 3-83
 DSK:, 2-35, 2-62
 DSKAS Device-related JSYSs, 3-97
 DSKAS JSYS, 3-97
 DSKOP Device-related JSYSs, 3-98
 DSKOP JSYS, 3-98
 DTACH Job-related JSYSs, 3-101
 DTACH JSYS, 3-101
 DTACH Terminal-related JSYSs, 3-101
 DTI JSYS, 3-101
 DTI Software-interrupt JSYSs, 3-101
 DTI Terminal-related JSYSs, 3-101
 Dump
 manipulating a, 3-94, 3-186
 Dump input, 3-102
 Dump mode, 2-46
 Dump of BUGCHK, 3-94, 3-186
 Dump output, 3-103
 DUMPI Dump-I/O JSYSs, 3-102
 DUMPI I/O JSYSs, 3-102
 DUMPI JSYS, 2-44, 3-102
 DUMPO Dump-I/O JSYSs, 3-103
 DUMPO I/O JSYSs, 3-103
 DUMPO JSYS, 3-103
 DVCHR Device-related JSYSs, 3-105
 DVCHR Info-returning JSYSs, 3-105
 DVCHR JSYS, 3-105

-E-

EJSERR macro, 1-14
 EJSHLT macro, 1-14
 Elapsed time process blocking,
 3-507
 Enabling capabilities, 3-117
 Enabling software interrupt
 system, 3-106
 End-of-file limit, 2-23
 ENQ ENQ/DEQ JSYSSs, 3-106
 ENQ JSYS, 3-106
 ENQC ENQ/DEQ JSYSSs, 3-113
 ENQC JSYS, 3-113
 Entering MDDT, 3-227
 Entry vector, 2-84
 process, 3-438
 EOF limit, 2-23
 EPCAP JSYS, 3-117
 EPCAP Parameter-setting JSYSSs,
 3-117
 EPCAP Process-related JSYSSs,
 3-117
 ERCAL, 2-23
 ERJMP, 2-23
 Error messages, 2-26
 Error strings, 2-26
 ERSTR Error-processing JSYSSs,
 3-118
 ERSTR JSYS, 3-118
 ERSTR% JSYS, 1-13
 ESOUT Error-processing JSYSSs,
 3-118
 ESOUT JSYS, 3-118
 Ethernet interface, 3-302
 Ethernet Loopback Operations,
 3-216
 Execute access, 2-9, 3-340
 Execute-only, 2-10
 Execute-only access, 2-9
 Execute-only files, 2-78
 Execute-only processes, 2-78
 Expunging files, 3-76

-F-

FDB, 2-11
 Attributes
 after RENAME, 3-395
 FE:, 2-35
 FFFFP File-related JSYSSs, 3-119
 FFFFP JSYS, 3-119
 FFFFP Page-related JSYSSs, 3-119
 FFORK JSYS, 3-119

FFORK Process-related JSYSSs,
 3-119
 FFUFP File-related JSYSSs, 3-120
 FFUFP JSYS, 3-120
 FFUFP Page-related JSYSSs, 3-120
 FH%EPN, 1-11
 .FHINF, 1-11
 .FHJOB, 1-11
 .FHSAI, 1-11
 .FHSLF, 1-11
 .FHSUP, 1-11
 .FHTOP, 1-11
 File
 date and time
 setting, 3-391
 File access, 2-9, 3-340
 File byte count, 2-22
 File date/time, 3-390
 File descriptor block, 2-11
 File designator, 1-8
 File functions, 2-1
 File handle, 2-3
 File handle indexable, 2-4
 File number job, 1-6
 File number job indexable, 1-6
 File opening a, 2-6
 File recognition, 2-2
 File specification, 2-1
 File status, 3-177
 File-archival functions, 2-92
 Files opening, 3-339
 Finding 1'st free file page,
 3-119
 Finding 1'st used file page,
 3-120
 FLIN I/O JSYSSs, 3-120
 FLIN JSYS, 3-120
 FLIN Numeric-I/O JSYSSs, 3-120
 FLIN TTY-I/O JSYSSs, 3-120
 Floating-point input, 3-120
 Floating-point output, 3-121
 FLOUT I/O JSYSSs, 3-121
 FLOUT JSYS, 3-121
 FLOUT Numeric-I/O JSYSSs, 3-121
 FLOUT TTY-I/O JSYSSs, 3-121
 Freezing a process, 3-119
 Frozen process, 3-385
 Full duplex mode, 2-49, 2-52
 Functions access-control, 2-74
 Functions accounting, 2-1
 Functions capabilities, 2-72
 Functions data-conversion, 2-86

Functions Date-and-Time, 2-89
 Functions device, 2-6, 2-34
 Functions file, 2-1
 Functions file-archival, 2-92
 Functions I/O, 2-22
 Functions I/O format-controlling,
 2-86
 Functions information-obtaining,
 2-26
 Functions line printer, 2-38
 Functions magnetic tape, 2-42
 Functions privileged, 2-94
 Functions process-control, 2-72
 Functions process-controlling,
 2-76
 Functions PSI, 2-64
 Functions software interrupt,
 2-64
 Functions terminal, 2-48

-G-

GACCT Accounting JSYSSs, 3-122
 GACCT Info-returning JSYSSs, 3-122
 GACCT Job-related JSYSSs, 3-122
 GACCT JSYS, 3-122
 GACTF Accounting JSYSSs, 3-122
 GACTF File-related JSYSSs, 3-122
 GACTF Info-returning JSYSSs, 3-122
 GACTF JSYS, 3-122
 Gaining directory access, 3-1
 GCVEC Info-returning JSYSSs, 3-123
 GCVEC JSYS, 3-123
 GDSKC Device-related JSYSSs, 3-123
 GDSKC Info-returning JSYSSs, 3-123
 GDSKC JSYS, 3-123
 GDSTS Device-related JSYSSs, 3-124
 GDSTS Info-returning JSYSSs, 3-124
 GDSTS JSYS, 3-124
 GDVEC Info-returning JSYSSs, 3-125
 GDVEC JSYS, 3-125
 GET JSYS, 3-125
 GET Page-related JSYSSs, 3-125
 GET Process-related JSYSSs, 3-125
 GETAB Info-returning JSYSSs, 3-128
 GETAB JSYS, 3-128
 GETER Error-processing JSYSSs,
 3-129
 GETER Info-returning JSYSSs, 3-129
 GETER JSYS, 3-129
 GETER% JSYS, 1-13
 GETJI Info-returning JSYSSs, 3-129

GETJI Job-related JSYSSs, 3-129
 GETJI JSYS, 3-129
 GETNM Info-returning JSYSSs, 3-131
 GETNM Job-related JSYSSs, 3-131
 GETNM JSYS, 3-131
 GETOK JSYS, 3-461
 GETOK% Access-control JSYSSs,
 3-132
 GETOK% Info-returning JSYSSs,
 3-132
 GETOK% JSYS, 3-132
 Getting a fork handle, 3-142
 Getting a save file, 3-125
 GEVEC Info-returning JSYSSs, 3-142
 GEVEC JSYS, 3-142
 GEVEC Process-related JSYSSs,
 3-142
 GFRKH Info-returning JSYSSs, 3-142
 GFRKH JSYS, 3-142
 GFRKH Process-related JSYSSs,
 3-142
 GFRKS Job-related JSYSSs, 3-143
 GFRKS JSYS, 3-143
 GFRKS Process-related JSYSSs,
 3-143
 GFUST File-related JSYSSs, 3-145
 GFUST Info-returning JSYSSs, 3-145
 GFUST JSYS, 3-145
 GIVOK% Access-control JSYSSs,
 3-146
 GIVOK% JSYS, 3-146
 GJINF Directory-related JSYSSs,
 3-146
 GJINF Info-returning JSYSSs, 3-146
 GJINF Job-related JSYSSs, 3-146
 GJINF JSYS, 3-146
 GJINF Terminal-related JSYSSs,
 3-146
 Global address, 1-3
 Global page numbers, 1-4
 GNJFN Directory-related JSYSSs,
 3-147
 GNJFN File-related JSYSSs, 3-147
 GNJFN JSYS, 3-147
 GNJFN Structure-related JSYSSs,
 3-147
 GPJFN Info-returning JSYSSs, 3-148
 GPJFN JSYS, 3-148
 GPJFN Process-related JSYSSs,
 3-148
 Greenwich Mean Time, 1-12
 GTAD Date/time JSYSSs, 3-148

GTAD Info-returning JSYSSs, 3-148
 GTAD JSYS, 3-148
 GTDAL Device-related JSYSSs, 3-149
 GTDAL Info-returning JSYSSs, 3-149
 GTDAL JSYS, 3-149
 GTDIR Directory-related JSYSSs,
 3-149
 GTDIR Info-returning JSYSSs, 3-149
 GTDIR JSYS, 3-149
 GTFDB File-related JSYSSs, 3-151
 GTFDB Info-returning JSYSSs, 3-151
 GTFDB JSYS, 3-151
 GTHST% JSYS, 3-151
 GTHST% TCP/IP-related JSYSSs,
 3-151
 GTJFN JSYS, 3-159, 3-167
 GTJFN(long) File-related JSYSSs,
 3-167
 GTJFN(short) File-related JSYSSs,
 3-159
 GTRPI Info-returning JSYSSs, 3-175
 GTRPI JSYS, 3-175
 GTRPI Page-related JSYSSs, 3-175,
 3-349
 GTRPI Process-related JSYSSs,
 3-175, 3-349
 GTRPI Trap-related JSYSSs, 3-175
 GTRPW Info-returning JSYSSs, 3-176
 GTRPW JSYS, 3-176
 GTRPW Trap-related JSYSSs, 3-176
 GTSTS File-related JSYSSs, 3-177
 GTSTS Info-returning JSYSSs, 3-177
 GTSTS JSYS, 3-177
 GTSTS Parameter-reading JSYSSs,
 3-177
 GTTYP Info-returning JSYSSs, 3-178
 GTTYP JSYS, 3-178
 GTTYP Parameter-reading JSYSSs,
 3-178
 GTTYP Terminal-related JSYSSs,
 3-178

-H-

Half duplex mode, 2-49, 2-52
 HALTF JSYS, 3-178
 HALTF Process-related JSYSSs,
 3-178
 Halting a process, 3-178, 3-179
 Halting system, 3-180
 Handle page, 3-394
 Handle process/file, 1-11

Handle section, 3-400
 Hardware data modes, 2-45
 HFORK JSYS, 3-179
 HFORK Process-related JSYSSs,
 3-179
 High density mode, 2-47
 Hostname, 3-330
 HPTIM Clock-related JSYSSs, 3-179
 HPTIM Info-returning JSYSSs, 3-179
 HPTIM JSYS, 3-179
 HSYS JSYS, 3-180

-I-

I/O data conversion, 2-86
 I/O errors, 2-23
 I/O format control, 2-86, 2-87
 I/O format-controlling Functions,
 2-86
 I/O functions, 2-22
 I/O modes, 2-61
 IDCNV Date/time JSYSSs, 3-181
 IDCNV JSYS, 3-181
 IDTIM Date/time JSYSSs, 3-182
 IDTIM I/O JSYSSs, 3-182
 IDTIM JSYS, 3-182
 IDTIM TTY-I/O JSYSSs, 3-151, 3-182
 IDTNC Date/time JSYSSs, 3-184
 IDTNC I/O JSYSSs, 3-184
 IDTNC JSYS, 3-184
 IDTNC TTY-I/O JSYSSs, 3-184
 IIC JSYS, 3-186
 IIC Process-related JSYSSs, 3-186
 IIC Software-interrupt JSYSSs,
 3-186
 Immediate terminal interrupt,
 2-70
 Indexable file handle, 2-4
 Indexable JFN, 1-6
 Indexable job file number, 1-6
 Industry compatible mode, 2-46
 INFO% interface, 3-186
 INFO% JSYS, 3-186
 Information
 configuration
 CNFIG% JSYS, 3-34
 Information-obtaining functions,
 2-26
 Initializing a process, 3-381
 Initiating software interrupts,
 3-186
 INLNM Info-returning JSYSSs, 3-195

INLNM JSYS, 3-195
INLNM Logical-name JSYSs, 3-195
Inputting a number, 3-320
Inputting date/time, 3-182, 3-184
Interface
 Ethernet, 3-302
Internal date/time format, 1-11
Internet datagram
 receiving, 3-375
Internet protocol
 IPOPR JSYS, 3-196
Internet queue
 release ownership, 3-380
Internet transmission, 2-58
Interrupt channel activation, 3-7
IPCF logout message, 3-72
IPOPR JSYS, 3-196

-J-

JFN, 1-6, 3-147, 3-148, 3-159,
 3-167, 3-177, 3-197
JFN mode word, 2-49, 3-384
JFN Status Word, 3-177
JFNS File-related JSYSs, 3-197
JFNS Info-returning JSYSs, 3-197
JFNS JSYS, 3-197
Job capabilities, 2-72
Job file number, 1-6
Job parameters, 3-428
Job Storage Block, 3-398
JSYS
 ERSTR%, 1-13
 GETER%, 1-13
JSYS arguments, 1-2
JSYS return, 1-1
JSYSs
 ASNIQ%, 3-13

-K-

KFORK JSYS, 3-200
KFORK Process-related JSYSs,
 3-200
Killing a process, 3-200

-L-

LATOP% JSYS, 3-200
LGOUT Job-related JSYSs, 3-215
LGOUT JSYS, 3-215
Line printer functions, 2-38

Line sequence numbers, 3-436
LLMOP% JSYS, 3-216
LNMST JSYS, 3-224
Loading VFU, 3-226
Local Area Terminals, 3-200
local byte pointer, 1-5
Local time, 1-12
Logging in a job, 3-225
Logical magnetic tape, 2-48
Logical name, 3-75
Logical name JSYS, 3-224
Logical names, 2-3
LOGIN Job-related JSYSs, 3-225
LOGIN JSYS, 3-225
Long form GTJFN, 3-167
Low Level Maintenance Operation,
 3-216
LPINI Device-related JSYSs, 3-226
LPINI JSYS, 3-226
LPT:, 2-35, 2-41, 2-62
LSN, 3-436

-M-

MACSYM macros, vi
 EJSERR, 1-14
 EJSHLT, 1-14
Magnetic tape functions, 2-42
Magnetic tape logical, 2-48
Magnetic tape physical, 2-48
Magnetic tape status bits, 2-48
Manipulating a spooled device,
 3-477
Manual pointer conventions, 1-9
Mapping a section, 3-455
Mapping memory, 3-400, 3-455
MDDT% Debugging JSYSs, 3-227
MDDT% JSYS, 3-227
Memory, 3-455
METER% Clock-related JSYSs, 3-227
METER% Info-returning JSYSs,
 3-227
METER% JSYS, 3-227
Modes access, 2-9
Modes I/O, 2-61
Modifying the FDB, 2-11
Monitor data base information,
 3-459, 3-511
Mountable-structure functions,
 3-236
MRECV IPCF JSYSs, 3-229
MRECV JSYS, 3-229

MRECV Process-related JSYSSs,
 3-229
 MSEND IPCF JSYSSs, 3-231
 MSEND JSYS, 3-231
 MSEND Process-related JSYSSs,
 3-231
 MSFRK JSYS, 3-235
 MSFRK Process-related JSYSSs,
 3-235
 .MSSSS, 3-236
 MSTR Info-returning JSYSSs, 3-236
 MSTR JSYS, 3-236
 MSTR Structure-related JSYSSs,
 3-236
 MT Device functions, 3-293
 MT:, 2-35, 2-48, 2-62
 MTA:, 2-35, 2-42, 2-62
 MTA: status bits, 2-42
 MTALN Device-related JSYSSs, 3-257
 MTALN JSYS, 3-257
 MTOPR Device-related JSYSSs, 3-257
 MTOPR JSYS, 3-257
 MTU% Device-related JSYSSs, 3-293
 MTU% JSYS, 3-293
 MUTIL IPCF JSYSSs, 3-295
 MUTIL JSYS, 3-295

-N-

Network Functions, 3-321, 3-332
 Network information, 3-330
 Network management operations,
 3-196
 NI Remote Console Service, 3-216
 NI% JSYS, 3-302
 NIN I/O JSYSSs, 3-320
 NIN JSYS, 3-320
 NIN Numeric-I/O JSYSSs, 3-320
 NIN TTY-I/O JSYSSs, 3-320
 NODE JSYS, 3-321
 Nonsharable save file, 2-80
 Nonshareable save, 3-405
 NOUT I/O JSYSSs, 3-329
 NOUT JSYS, 3-329
 NOUT Numeric-I/O JSYSSs, 3-329
 NOUT TTY-I/O JSYSSs, 3-329
 NTINF% JSYS, 3-330
 NTMAN% JSYS, 3-332
 NUL:, 2-35, 2-63
 Numbers
 line sequence, 3-436

-O-

Obtaining interrupt table
 addresses, 3-531
 ODCNV Date/time JSYSSs, 3-334
 ODCNV JSYS, 3-334
 ODTIM Date/time JSYSSs, 3-336
 ODTIM JSYS, 3-336
 ODTNC Date/time JSYSSs, 3-338
 ODTNC JSYS, 3-338
 Offline expiration date and time,
 3-390
 One-word global byte pointer, 1-5
 Online expiration date and time,
 3-390
 OPENF File-related JSYSSs, 3-339
 OPENF JSYS, 3-339
 Opening a device, 2-6
 Opening a file, 2-6, 3-339
 Opening files, 3-339
 Outputting a number, 3-329
 Outputting date/time, 3-336,
 3-338
 Outputting error strings, 3-118
 Overflow trapping, 3-491

-P-

PA1050, 2-80
 Page access, 3-394, 3-531
 Page handle, 3-394
 Page mapping, 3-350, 3-352, 3-353,
 3-354
 Panic channels, 2-65, 2-67
 Parse-only file specification,
 2-5
 Parse-only JFN, 3-173, 3-199
 PBIN Byte-I/O JSYSSs, 3-344
 PBIN I/O JSYSSs, 3-344
 PBIN JSYS, 3-344
 PBIN TTY-I/O JSYSSs, 3-344
 PBOUT Byte-I/O JSYSSs, 3-345
 PBOUT I/O JSYSSs, 3-345
 PBOUT JSYS, 3-345
 PBOUT TTY-I/O JSYSSs, 3-345
 PC histogram, 3-466
 PCDP, 2-35
 PCDP:, 2-35, 2-37, 2-61
 PCDP: status bits, 2-37
 PCDR:, 2-35, 2-36, 2-61
 PCDR: status bits, 2-36
 PDVOP% JSYS, 3-345

PEEK Debugging JSYSs, 3-348
 PEEK JSYS, 3-348
 Physical magnetic tape, 2-48
 Physical/logical tape-drive association, 3-257
 PLOCK JSYS, 3-349
 PLPT:, 2-35, 2-38, 2-40, 2-62
 PLPT: control characters, 2-39
 PLPT: status bits, 2-40
 PMAP File-related JSYSs, 3-350
 PMAP JSYS, 3-350
 PMAP Page-related JSYSs, 3-350
 PMAP Process-related JSYSs, 3-350
 PMCTL JSYS, 3-355
 PMCTL Page-related JSYSs, 3-355
 PPN, 3-358, 3-488
 PPNST Info-returning JSYSs, 3-358
 PPNST JSYS, 3-358
 PRARG Info-returning JSYSs, 3-359
 PRARG JSYS, 3-359
 PRARG Parameter-reading JSYSs, 3-359
 PRARG Parameter-setting JSYSs, 3-359
 PRARG Process-related JSYSs, 3-359
 Primary input designator, 3-377
 Primary input file, 2-22
 Primary output designator, 3-377
 Primary output file, 2-22
 Print request, 3-361
 Private program name, 3-431
 Privileged functions, 2-94
 Privileged monitor calls, 2-94
 Process capabilities, 2-72
 Process entry vector, 3-438
 Process handle, 1-10
 Process handle relative, 1-11
 Process operations, 2-76, 3-475, 3-515, 3-524
 Process status, 3-388
 Process timing, 3-507
 Process-control functions, 2-72
 Process-controlling functions, 2-76
 Process/file handle, 1-11
 Program data vector, 3-345
 Program debugging, 3-3
 Project-programmer number (PPN), 3-358, 3-488
 PSI functions, 2-64
 PSOUT I/O JSYSs, 3-361
 PSOUT JSYS, 3-361
 PSOUT String-I/O JSYSs, 3-361
 PTY:, 2-35

-Q-

QUEUE% JSYS, 3-361

-R-

Random byte input, 3-391
 Random byte output, 3-396
 RCDIR Directory-related JSYSs, 3-368
 RCDIR Info-returning JSYSs, 3-368
 RCDIR JSYS, 3-368
 RCM Info-returning JSYSs, 3-372
 RCM JSYS, 3-372
 RCM Software-interrupt JSYSs, 3-372
 RCUSR Info-returning JSYSs, 3-372
 RCUSR JSYS, 3-372
 RCVIM JSYS, 3-374
 RCVIM TCP/IP-related JSYSs, 3-374
 RCVIM% JSYS, 3-375
 RCVOK% Access-control JSYSs, 3-376
 RCVOK% JSYS, 3-376
 RDTTY I/O JSYSs, 3-377
 RDTTY JSYS, 3-377
 RDTTY Terminal-related JSYSs, 3-377
 RDTTY TTY-I/O JSYSs, 3-377
 Read access, 2-9, 3-340
 Reading the FDB, 2-11
 Recognition file, 2-2
 Redefining controlling terminal, 3-423
 Regulated structure, 2-6, 3-2, 3-236, 3-255, 3-517
 Relative process handle, 1-11
 RELD Device-related JSYSs, 3-380
 RELD JSYS, 3-380
 Releasing a JFN, 3-393
 Releasing a process handle, 3-386
 Releasing devices, 3-380
 Releasing working set, 3-404
 Relinquishing directory access, 3-1
 RELIQ% JSYS, 3-380
 RELSQ JSYS, 3-381
 RELSQ TCP/IP-related JSYSs, 3-381

Renaming a file, 3-394
Rescan buffer, 3-398
Reserving a channel, 3-84
RESET JSYS, 3-381
RESET Process-related JSYSSs,
3-381
Resetting a process, 3-381
Resetting file byte size, 3-433
Restricted JFN, 3-177, 3-489
Resuming a process, 3-385
Resuming process execution, 3-521
Retrieving an IPCF message, 3-229
Returning CCOC words, 3-383
Returning device characteristics,
3-105
Returning device status, 3-124
Returning directory information,
3-149
Returning disk allocation, 3-149
Returning EBOX/MBOX meter values,
3-227
Returning elapsed system restart
time, 3-507
Returning file author, 3-145
Returning file byte-size, 3-383
Returning file descriptor block,
3-151
Returning file specification,
3-197
Returning file status, 3-177
Returning file's account, 3-122
Returning high-precision clock,
3-179
Returning interrupt mask, 3-393,
3-403
Returning interrupt table, 3-392
Returning JFN mode word, 3-384
Returning job information, 3-129,
3-146
Returning most recent error,
3-129
Returning PA1050 entry vector,
3-123
Returning page trap information,
3-175
Returning process AC's, 3-382
Returning process entry vector,
3-142, 3-529
Returning Process status, 3-387
Returning program name, 3-131
Returning RMS entry vector, 3-125
Returning system table, 3-128,
3-493
Returning TCP/IP host information,
3-151
Returning trap words, 3-176,
3-528
RFACS Info-returning JSYSSs, 3-382
RFACS JSYS, 3-382
RFACS Process-related JSYSSs,
3-382
RFBSZ File-related JSYSSs, 3-383
RFBSZ JSYS, 3-383
RFCOC JSYS, 3-383
RFCOC Terminal-related JSYSSs,
3-383
RFCOC TTY-I/O JSYSSs, 3-383
RFMOD File-related JSYSSs, 3-384
RFMOD Info-returning JSYSSs, 3-384
RFMOD JSYS, 3-384
RFORK JSYS, 3-385
RFORK Process-related JSYSSs,
3-385
RFPOS Info-returning JSYSSs, 3-385
RFPOS JSYS, 3-385
RFPOS Terminal-related JSYSSs,
3-385
RFPOS TTY-I/O JSYSSs, 3-385
RFPTR File-related JSYSSs, 3-386
RFPTR Info-returning JSYSSs, 3-386
RFPTR JSYS, 3-386
RFRKH JSYS, 3-386
RFRKH Process-related JSYSSs,
3-386
RFSTS Info-returning JSYSSs, 3-387
RFSTS JSYS, 3-387
RFSTS Process-related JSYSSs,
3-387
RFTAD Archive-related JSYSSs,
3-390
RFTAD Date/time JSYSSs, 3-390
RFTAD File-related JSYSSs, 3-390
RFTAD Info-returning JSYSSs, 3-390
RFTAD JSYS, 3-390
RIN Byte-I/O JSYSSs, 3-391, 3-396
RIN I/O JSYSSs, 3-391, 3-396
RIN JSYS, 3-391
RIN Random-I/O JSYSSs, 3-391,
3-396
RIR Info-returning JSYSSs, 3-392
RIR JSYS, 3-392
RIR Software-interrupt JSYSSs,
3-392

RIRCM Info-returning JSYSs, 3-393
RIRCM JSYS, 3-393
RIRCM Software-interrupt JSYSs,
3-393
RLJFN File-related JSYSs, 3-393
RLJFN JSYS, 3-393
RMAP JSYS, 3-394
RMAP Page-related JSYSs, 3-394
RMAP Process-related JSYSs, 3-394
RMS entry vector, 3-426
RNAME File-related JSYSs, 3-394
RNAME JSYS, 3-394
ROUT JSYS, 3-396
RPACS Info-returning JSYSs, 3-397
RPACS JSYS, 3-397
RPACS Page-related JSYSs, 3-397
RPCAP Info-returning JSYSs, 3-398
RPCAP JSYS, 3-398
RSCAN JSYS, 3-398
RSCAN Terminal-related JSYSs,
3-398
RSMAP% Info-returning JSYSs,
3-400
RSMAP% JSYS, 3-400
RSMAP% Process-related JSYSs,
3-400
RTFRK Info-returning JSYSs, 3-401
RTFRK JSYS, 3-401
RTFRK Process-related JSYSs,
3-401
RTIW Info-returning JSYSs, 3-402
RTIW JSYS, 3-402
RTIW Terminal-related JSYSs,
3-402
Run time, 3-402
RUNTM Info-returning JSYSs, 3-402
RUNTM JSYS, 3-402
RWM Info-returning JSYSs, 3-403
RWM JSYS, 3-403
RWM Process-related JSYSs, 3-403
RWM Software-interrupt JSYSs,
3-403
RWSET JSYS, 3-404
RWSET Page-related JSYSs, 3-404

-S-

SACTF Accounting JSYSs, 3-404
SACTF JSYS, 3-404
Sample program, 2-6
Save files, 2-80
SAVE JSYS, 3-405

SAVE Page-related JSYSs, 3-405
Scheduler control, 3-449
Scheduler priority control word,
3-448
SCS% JSYS, 3-406
SCTTY JSYS, 3-423
SCTTY Process-related JSYSs,
3-423
SCTTY Terminal-related JSYSs,
3-423
SCVEC JSYS, 3-424
SDSTS Device-related JSYSs, 3-426
SDSTS JSYS, 3-426
SDVEC JSYS, 3-426
Section handle, 3-400
Section mapping, 3-400
Section-relative address, 1-3
Section-relative page number, 1-4
Sending an IPCF message, 3-231
SETER Error-processing JSYSs,
3-427
SETER JSYS, 3-427
SETER Process-related JSYSs,
3-427
SETJB Job-related JSYSs, 3-428
SETJB JSYS, 3-428
SETJB Parameter-setting JSYSs,
3-428
SETNM Job-related JSYSs, 3-431
SETNM JSYS, 3-431
SETSN Job-related JSYSs, 3-431
SETSN JSYS, 3-431
Setting CCOC words, 3-433
Setting device mode, 3-487
Setting error condition, 3-427
Setting file author, 3-441
Setting file date/time, 3-439
Setting file pointer, 3-436
Setting file status, 3-489
Setting interrupt mask, 3-447
Setting interrupt table addresses,
3-446, 3-533
Setting job priority, 3-448
Setting monitor flags, 3-459
Setting page accessibility, 3-473
Setting primary JFN, 3-474
Setting process AC's, 3-432
Setting process entry vector,
3-431, 3-535
Setting process priority, 3-479
Setting program name, 3-431
Setting system date, 3-482

Setting terminal interrupt word, 3-485
 Setting terminal modes, 3-434
 Setting terminal number, 3-490
 Setting terminal pointer, 3-436
 SEVEC JSYS, 3-431
 SEVEC Process-related JSYSs, 3-431
 SFACS JSYS, 3-432
 SFACS Process-related JSYSs, 3-432
 SFBSZ File-related JSYSs, 3-433
 SFBSZ JSYS, 3-433
 SFCOC JSYS, 3-433
 SFCOC Terminal-related JSYSs, 3-433
 SFCOC TTY-I/O JSYSs, 3-433
 SFMOD JSYS, 3-434
 SFMOD Terminal-related JSYSs, 3-434
 SFMOD TTY-I/O JSYSs, 3-434
 SFORK JSYS, 3-435
 SFORK Process-related JSYSs, 3-435
 SFPOS JSYS, 3-436
 SFPOS Terminal-related JSYSs, 3-436
 SFPOS TTY-I/O JSYSs, 3-436
 SFPTR File-related JSYSs, 3-436
 SFPTR I/O JSYSs, 3-436
 SFPTR JSYS, 3-436
 SFRKV JSYS, 3-438
 SFRKV Process-related JSYSs, 3-438
 SFTAD Date/time JSYSs, 3-439
 SFTAD File-related JSYSs, 3-439
 SFTAD JSYS, 3-439
 SFUST File-related JSYSs, 3-441
 SFUST JSYS, 3-441
 Sharable save, 3-480
 Sharable save file, 2-81
 Short form GTJFN, 3-159
 SIBE File-related JSYSs, 3-442
 SIBE I/O JSYSs, 3-442
 SIBE JSYS, 3-442
 Simulating terminal input, 3-484
 Simulating terminal output, 3-486
 SIN I/O JSYSs, 3-443
 SIN JSYS, 3-443
 SIN String-I/O JSYSs, 3-443
 SIN TTY-I/O JSYSs, 3-443
 SINR I/O JSYSs, 3-444
 SINR JSYS, 3-444
 SINR Record-I/O JSYSs, 3-444
 SINR TTY-I/O JSYSs, 3-444
 SIR JSYS, 3-446
 SIR Process-related JSYSs, 3-446
 SIR Software-interrupt JSYSs, 3-446
 SIRCM JSYS, 3-447
 SIRCM Process-related JSYSs, 3-447
 SIRCM Software-interrupt JSYSs, 3-447
 SIXBIT mode, 2-47
 SIZEF File-related JSYSs, 3-447
 SIZEF Info-returning JSYSs, 3-447
 SIZEF JSYS, 3-447
 SJPRI Job-related JSYSs, 3-448
 SJPRI JSYS, 3-448
 SKED% JSYS, 3-449
 SKPIR JSYS, 3-455
 SKPIR Process-related JSYSs, 3-455
 SKPIR Software-interrupt JSYSs, 3-455
 SMAP% JSYS, 3-455
 SMON JSYS, 3-459
 SMON Parameter-setting JSYSs, 3-459
 SNDIM JSYS, 3-464
 SNDIM TCP/IP-related JSYSs, 3-464
 SNDIN% JSYS, 3-465
 SNOOP Debugging JSYSs, 3-466
 SNOOP JSYS, 3-466
 SOBE File-related JSYSs, 3-469
 SOBE I/O JSYSs, 3-469
 SOBE JSYS, 3-469
 SOBF File-related JSYSs, 3-470
 SOBF I/O JSYSs, 3-470
 SOBF JSYS, 3-470
 Software data modes, 2-61, 2-62, 2-64
 Software interrupt channel, 2-64
 Software interrupt functions, 2-64
 Software interrupt priority, 2-66
 Software interrupt system, 2-64
 Software interrupt table, 2-66
 Source designator, 1-6
 Source/destination designator, 1-6
 SOUT I/O JSYSs, 3-470
 SOUT JSYS, 3-470

SOUT String-I/O JSYSs, 3-470
 SOUT TTY-I/O JSYSs, 3-470
 SOUTR I/O JSYSs, 3-472
 SOUTR JSYS, 3-472
 SOUTR Record-I/O JSYSs, 3-472
 SPACS JSYS, 3-473
 SPACS Page-related JSYSs, 3-473
 SPJFN File-related JSYSs, 3-474
 SPJFN JSYS, 3-474
 SPJFN Process-related JSYSs,
 3-474
 SPLFK JSYS, 3-475
 SPLFK Process-related JSYSs,
 3-475
 Splicing a process, 3-475
 SPOOL Directory-related JSYSs,
 3-477
 SPOOL JSYS, 3-477
 SPRIW JSYS, 3-479
 SPRIW Process-related JSYSs,
 3-479
 SSAVE JSYS, 3-480
 SSAVE Process-related JSYSs,
 3-480
 STAD Date/time JSYSs, 3-482
 STAD JSYS, 3-482
 Standard date/time, 1-11
 Starting a process, 3-235, 3-435,
 3-438, 3-533
 Status bits magnetic tape, 2-48
 Status bits terminal, 2-48
 STCMP JSYS, 3-482
 STCMP String-compare JSYSs, 3-482
 STDEV Directory-related JSYSs,
 3-483
 STDEV Info-returning JSYSs, 3-483
 STDEV JSYS, 1-10, 3-483
 STI JSYS, 3-484
 STI Terminal-related JSYSs, 3-484
 STI TTY-I/O JSYSs, 3-484
 STIW JSYS, 3-485
 STIW Software-interrupt JSYSs,
 3-485
 STIW Terminal-related JSYSs,
 3-485
 STO JSYS, 3-486
 STO Terminal-related JSYSs, 3-486
 STO TTY-I/O JSYSs, 3-486
 STPAR Device-related JSYSs, 3-487
 STPAR JSYS, 3-487
 STPPN Directory-related JSYSs,
 3-488
 STPPN JSYS, 3-488
 String input, 3-443
 String output, 3-361, 3-470
 Strings, 1-8
 STSTS File-related JSYSs, 3-489
 STSTS JSYS, 3-489
 STTYP JSYS, 3-490
 STTYP Terminal-related JSYSs,
 3-490
 Swapping JFNs, 3-490
 SWJFN File-related JSYSs, 3-490
 SWJFN JSYS, 3-490
 SWTRP% Debugging JSYSs, 3-491
 SWTRP% JSYS, 3-491
 SYERR Error-processing JSYSs,
 3-492
 SYERR JSYS, 3-492
 SYSGT Info-returning JSYSs, 3-493
 SYSGT JSYS, 3-493
 SYSTAT table, 2-31
 System accounting, 3-516
 System date, 1-11, 2-89
 System error file, 3-492
 System Message Levels, 3-460
 System performance analysis,
 3-466
 System program name, 3-431
 System tables, 2-27

-T-

Table searching, 3-495
 TBADD JSYS, 3-493
 TBADD Table-lookup JSYSs, 3-493
 TBDEL JSYS, 3-494
 TBDEL Table-lookup JSYSs, 3-494
 TBLUK JSYS, 3-495
 TBLUK Table-lookup JSYSs, 3-495
 TCOPR% JSYS, 3-497
 TCP:, 2-35, 2-58, 2-63
 GTJFN format, 2-58
 OPENF JSYS, 2-59
 other JSYSs, 2-60
 Terminal advising, 2-58
 Terminal data mode, 2-49, 2-51
 Terminal designator, 1-6, 1-10
 Terminal functions, 2-48
 Terminal interrupt, 2-67
 Terminal interrupt codes, 2-67
 Terminal interrupt modes, 2-70
 Terminal interrupt word, 3-402
 Terminal length, 2-49, 2-50

Terminal linking, 2-58, 3-509
Terminal status bits, 2-48
Terminal type
 Returning, 3-178
Terminal width, 2-49, 2-50
Testing file input buffer, 3-442
Testing file output buffer, 3-469, 3-470
Testing for end-of-file, 2-24
Testing monitor flags, 3-511
TEXTI I/O JSYSs, 3-499
TEXTI JSYS, 3-499
TEXTI Terminal-related JSYSs, 3-499
TEXTI TTY-I/O JSYSs, 3-499
TFORK JSYS, 3-504
TFORK Process-related JSYSs, 3-504
Thawed access, 2-9, 3-340
THIBR JSYS, 3-507
THIBR Process-related JSYSs, 3-507
TIME JSYS, 3-507
Time zone, 1-12
TIMER JSYS, 3-507
TIMER Process-related JSYSs, 3-507
TIMEZONE offset, 1-12
TLINK JSYS, 3-509
TLINK Terminal-related JSYSs, 3-509
TMON Info-returning JSYSs, 3-511
TMON JSYS, 3-511
TMON Parameter-reading JSYSs, 3-511
TOPS-10 monitor calls, 1-1
Translating a logical name, 3-224
Translating device name string, 3-483
Translating device string, 3-82
Translating directory name string, 3-488
Translating directory number, 3-93
Translating error strings, 3-118
Transmission control protocol, 2-58
TTMSG I/O JSYSs, 3-514
TTMSG JSYS, 3-514
TTMSG Terminal-related JSYSs, 3-514
TTMSG TTY-I/O JSYSs, 3-514
TTY:, 2-35, 2-48, 2-64
TTY: characteristics, 2-55, 2-56
TTY: status bits, 2-48
TWAKE JSYS, 3-515
TWAKE Process-related JSYSs, 3-515
Two-word global byte pointer, 1-5
Two-word local byte pointer, 1-5

-U-

UFPGS File-related JSYSs, 3-516
UFPGS JSYS, 3-516
UFPGS Page-related JSYSs, 3-516
Unbuffered I/O, 2-44
Underflow trapping, 3-491
Universal default designator, 1-9
Updating file pages, 3-516
USAGE Accounting JSYSs, 3-516
USAGE JSYS, 3-516
User-I/O mode, 3-520
USRIO I/O JSYSs, 3-520
USRIO JSYS, 3-520
UTEST Debugging JSYSs, 3-520
UTEST JSYS, 3-520
UTFRK JSYS, 3-521
UTFRK Process-related JSYSs, 3-521
UUO's, 1-1

-V-

VACCT JSYS, 3-523
Verifying accounts, 3-523
VFU, 2-39

-W-

WAIT JSYS, 3-523
WAIT Process-related JSYSs, 3-523
Waiting for process termination, 3-524
Wakeup class, 2-52, 2-54
Waking a process, 3-515
WFORK JSYS, 3-524
WFORK Process-related JSYSs, 3-524
Wild string comparison, 3-524
WILD% JSYS, 3-524
WILD% String-compare JSYSs, 3-524
Wildcard characters, 2-4
Working Set Management, 3-526

Write access, 2-9, 3-340
Write-to-operator, 3-361
WSMGR% JSYS, 3-526

-X-

XGTPW% Info-returning JSYSs,
3-528
XGTPW% JSYS, 3-528
XGTPW% Trap-related JSYSs, 3-528
XGVEC% Info-returning JSYSs,
3-529
XGVEC% JSYS, 3-529
XGVEC% Process-related JSYSs,
3-529
XRIR% JSYS, 3-531
XRIR% Process-related JSYSs,
3-531

XRIR% Software-interrupt JSYSs,
3-531
XRMAP% JSYS, 3-531
XRMAP% Page-related JSYSs, 3-531
XRMAP% Process-related JSYSs,
3-531
XSFRK% JSYS, 3-533
XSFRK% Process-related JSYSs,
3-533
XSIR% JSYS, 3-533
XSIR% Process-related JSYSs,
3-533
XSIR% Software-interrupt JSYSs,
3-533
XSVEC% JSYS, 3-535
XSVEC% Process-related JSYSs,
3-535